



KEPUTUSAN REKTOR UNIVERSITAS MULAWARMAN

NOMOR : 547 /UN17/HK/2022

TENTANG

PENGANGKATAN DOSEN PENGAMPU MATA KULIAH
PROGRAM MERDEKA BELAJAR KAMPUS MERDEKA (MBKM)
SEMESTER GENAP TAHUN AKADEMIK 2021/2022
PADA PROGRAM STUDI S1 STATISTIKA, S1 MATEMATIKA,
S1 BIOLOGI, S1 KIMIA, S1 FISIKA DAN S1 GEOFISIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS MULAWARMAN

REKTOR UNIVERSITAS MULAWARMAN,

- Menimbang : a. bahwa untuk menjamin kepastian hukum dalam rangka tertib administrasi dan kelancaran pelaksanaan kegiatan belajar mengajar Program Merdeka Belajar Kampus Merdeka (MBKM) Semester Genap Tahun Akademik 2021/2022 pada Program Studi S1 Statistika, S1 Matematika, S1 Biologi, S1 Kimia, S1 Fisika dan S1 Geofisika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Mulawarman, maka dipandang perlu mengangkat Dosen Pengampu Mata Kuliah;
- b. bahwa berdasarkan pertimbangan sebagaimana dimaksud dalam huruf a, perlu menetapkan Keputusan Rektor Universitas Mulawarman tentang Pengangkatan Dosen Pengampu Mata Kuliah Program Merdeka Belajar Kampus Merdeka (MBKM) Semester Genap Tahun Akademik 2021/2022 pada Program Studi S1 Statistika, S1 Matematika, S1 Biologi, S1 Kimia, S1 Fisika dan S1 Geofisika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Mulawarman;
- Mengingat : 1. Undang-Undang RI Nomor 20 Tahun 2003 tentang Sistem Pendidikan Nasional;
2. Undang-Undang RI Nomor 12 tahun 2012 tentang Pendidikan Tinggi;
3. Undang-Undang RI Nomor 5 tahun 2014 tentang Aparatur Sipil Negara;
4. Peraturan Pemerintah RI Nomor 23 Tahun 2005 tentang Pengelolaan Keuangan Badan Layanan Umum, sebagaimana telah diubah dengan Peraturan Pemerintah RI Nomor 74 Tahun 2012 tentang Perubahan Atas Peraturan Pemerintah RI Nomor 23 Tahun 2005 tentang Pengelolaan Keuangan Badan Layanan Umum;
5. Peraturan Pemerintah RI Nomor 37 Tahun 2009 tentang Dosen;
6. Peraturan Pemerintah RI Nomor 4 Tahun 2014 tentang Penyelenggaraan Pendidikan Tinggi dan Pengelolaan Perguruan Tinggi;
7. Keputusan Presiden RI Nomor 65 Tahun 1963 tentang Pendirian Universitas Mulawarman;
8. Peraturan Menteri Riset, Teknologi dan Pendidikan Tinggi RI Nomor 9 Tahun 2015 tentang Organisasi dan Tata Kerja Universitas Mulawarman, sebagaimana telah diubah dengan Peraturan Menteri Riset, Teknologi dan Pendidikan Tinggi RI Nomor 26 Tahun 2018 tentang Perubahan Atas Peraturan Menteri Riset, Teknologi dan Pendidikan Tinggi RI Nomor 9 Tahun 2015 tentang Organisasi dan Tata Kerja Universitas Mulawarman;

9. Peraturan Menteri Riset, Teknologi dan Pendidikan Tinggi RI Nomor 57 Tahun 2018 tentang Statuta Universitas Mulawarman;
10. Keputusan Menteri Keuangan RI Nomor 51/KMK.05/2009 tentang Penetapan Universitas Mulawarman sebagai Instansi Pemerintah yang Menerapkan Pengelolaan Keuangan Badan Layanan Umum;
11. Keputusan Menteri Riset, Teknologi, dan Pendidikan Tinggi RI Nomor 661/M/KPT.KP/2018 tentang Pemberhentian dan Pengangkatan Rektor Universitas Mulawarman Periode Tahun 2018-2022;
12. Peraturan Rektor Universitas Mulawarman Nomor 17 Tahun 2020 tentang Penyelenggaraan Pendidikan dan Pengajaran, Penelitian dan Pengabdian Kepada Masyarakat Berbasis Kampus Merdeka dan Merdeka Belajar;
13. Keputusan Rektor Universitas Mulawarman Nomor 109/OT/2006 Tahun 2006 tentang Peningkatan Status Unit Pelaksana FMIPA Menjadi Fakultas Matematika dan Ilmu Pengetahuan Alam (FMIPA) Universitas Mulawarman;
14. Keputusan Rektor Universitas Mulawarman Nomor 2414/KP2018 tentang Pemberhentian dan Pengangkatan Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Mulawarman Periode 2018-2022;

Memperhatikan : Surat Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Mulawarman Nomor 51/UN17.7/TU/2022 tanggal 12 Januari 2022, perihal Permohonan Penerbitan SK Rektor.

MEMUTUSKAN:

- Menetapkan : KEPUTUSAN REKTOR UNIVERSITAS MULAWARMAN TENTANG PENGANGKATAN DOSEN PENGAMPU MATA KULIAH PROGRAM MERDEKA BELAJAR KAMPUS MERDEKA (MBKM) SEMESTER GENAP TAHUN AKADEMIK 2021/2022 PADA PROGRAM STUDI S1 STATISTIKA, S1 MATEMATIKA, S1 BIOLOGI, S1 KIMIA, S2 KIMIA, S1 FISIKA DAN S1 GEOFISIKA FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM UNIVERSITAS MULAWARMAN.
- KESATU : Mengangkat nama-nama yang tercantum dalam lampiran yang tidak terpisahkan dari Keputusan ini sebagai Dosen Pengampu Mata Kuliah Program Merdeka Belajar Kampus Merdeka (MBKM) Semester Genap Tahun Akademik 2021/2022 pada Program Studi S1 Statistika, S1 Matematika, S1 Biologi, S1 Kimia, S1 Fisika dan S1 Geofisika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Mulawarman.
- KEDUA : Dosen Pengampu Mata Kuliah Program Merdeka Belajar Kampus Merdeka (MBKM) sebagaimana dimaksud pada diktum kesatu keputusan ini dalam melaksanakan tugasnya bertanggung jawab kepada Rektor Universitas Mulawarman melalui Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Mulawarman.
- KETIGA : Pembiayaan akibat ditetapkannya keputusan ini dibebankan DIPA BLU Universitas Mulawarman, anggaran Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Mulawarman.
- KEEMPAT : Keputusan ini berlaku sejak tanggal 3 Januari 2022.
- KELIMA : Bilamana dikemudian hari terdapat kekeliruan dalam keputusan ini akan diubah dan diperbaiki sebagaimana mestinya.

Ditetapkan di Samarinda
pada tanggal 12 Januari 2022



H. Masjaya, M.Si.

NIP. 196212311991031024

LAMPIRAN I
KEPUTUSAN REKTOR UNIVERSITAS MULAWARMAN
NOMOR 547 /UN17/HK/2022
TANGGAL 12 JANUARI 2022
TENTANG
PENGANGKATAN DOSEN PENGAMPU MATA KULIAH
PROGRAM MERDEKA BELAJAR KAMPUS MERDEKA
(MBKM) SEMESTER GENAP TAHUN AKADEMIK
2021/2022 PADA PROGRAM STUDI S1 STATISTIKA,
S1 MATEMATIKA, S1 BIOLOGI, S1 KIMIA, S1 FISIKA
DAN S1 GEOFISIKA FAKULTAS MATEMATIKA DAN
ILMU PENGETAHUAN ALAM UNIVERSITAS
MULAWARMAN.

PROGRAM STUDI : S1-STATISTIKA

N O	KODE MK	MATA KULIAH	SKS/ SMT	JENIS MK (W/P)	KURIKULUM	DOSEN PENGAMPU MK
(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	210700603W005	Statistika Dasar	3/I	W	2021	Dr. Sifriyani, S.Pd., M.Si Ika Purnamasari, M. Si
2	190701603W018	Analisis Regresi	3/IV	W	2019	Dr. Darnah Andi Nohe, M.Si Dr. M. Fathurahman, M.Si
3	190701603W020	Pengantar Metode Survei	3/IV	W	2019	Memi Nor Hayati, S.Si, M.Si Dr. M. Fathurahman, M.Si
4	190701603W021	Rancangan Percobaan	3/IV	W	2019	Dr. M. Fathurahman, M.Si Memi Nor Hayati, S.Si, M.Si
5	190701603W023	Statistika Pengendalian Mutu	3/IV	W	2019	Dr. Sri Wahyuningsih, M. Si Meiliyani Siringoringo, S.Si., M.Si
6	190701603P026	Biostatistik	3/IV	P	2019	Dr. Darnah Andi Nohe, M.Si Memi Nor Hayati, S.Si, M.Si
7	190701603W035	Data Mining	3/VI	W	2019	Surya Prangga, S.Si., M.Si Rito Goejantoro, S.Si., M. Si
8	190701603W036	Analisis Data Eksploratif	3/VI	W	2019	Ika Purnamasari, S.Si, M.Si Dr. Darnah A. Nohe, M.Si
9	190701603W037	Pengantar Model Linier	3/VI	W	2019	Dr. Sifriyani, S.Pd., M.Si Meiliyani Siringoringo, M.Si
10	190701603P039	Teori Antrian	3/VI	P	2019	Dr. Sri Wahyuningsih, M.Si Dr. M. Fathurahman, M.Si
11	190701603P040	Analisis Data Uji Hidup	3/VI	P	2019	Dr. Suyitno, S.Pd., M.Sc Nariza Wanti Wulan Sari, S.Si., M.Si
12	190701603P041	Ekonometrika	3/VI	P	2019	Dr. Sifriyani, S.Pd., M.Si Dr. Darnah Andi Nohe, M.Si
13	190701603P042	Matematika Asuransi	3/VI	P	2019	Dr. Suyitno, S.Pd., M.Sc Dr. Sri Wahyuningsih, M.Si

Ditetapkan di Samarinda



H. Masjaya, M.Si.
NIP 196212311991031024

LAMPIRAN II

KEPUTUSAN REKTOR UNIVERSITAS MULAWARMAN

NOMOR 547 /UN17/HK/2022

TANGGAL 12 JANUARI 2022

TENTANG

PENGANGKATAN DOSEN PENGAMPU MATA KULIAH PROGRAM MERDEKA BELAJAR KAMPUS MERDEKA (MBKM) SEMESTER GENAP TAHUN AKADEMIK 2021/2022 PADA PROGRAM STUDI S1 STATISTIKA, S1 MATEMATIKA, S1 BIOLOGI, S1 KIMIA, S1 FISIKA DAN S1 GEOFISIKA FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM UNIVERSITAS MULAWARMAN.

MATRIKS MATA KULIAH PROGRAM MERDEKA BELAJAR KAMPUS MERDEKA (MBKM) SEMESTER GENAP TAHUN AKADEMIK 2021/2022 PROGRAM STUDI S1 STATISTIKA FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM UNIVERSITAS MULAWARMAN

Nama Dosen : Dr. Sri Wahyuningsih, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Statistika Pengendalian Mutu	W	Meiliyani Siringoringo, S.Si., M. Si	2	1	IV
2	Teori Antrian	P	Dr. M. Fathurahman, M.Si.	3	0	VI
3	Matematika Asuransi	P	Dr. Suyitno, S.Pd., M.Sc.	3	0	VI
Total				8	1	

Nama Dosen : Rita Goejantero, S.Si., M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Data Mining	W	Surya Prangga, S.Si, M.Si	2	1	VI
Total				2	1	

Nama Dosen : Dr. Darnah A. Nohe, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Analisis Regresi	W	Dr. M. Fathurahman, M.Si	2	1	IV
2	Analisis Data Eksploratif	W	Ika Purnamasari, S.Si, M.Si	2	1	VI
3	Ekonometrika	P	Dr. Sifriyani, S.Pd., M.Si	2	1	VI
4	Biostatistika	P	Memmi Nor Hayati, S.Si., M.Si	2	1	IV
Total				8	4	

Nama Dosen : Dr. Suyitno, S.Pd., M.Sc

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Matematika Asuransi	P	Dr. Sri Wahyuningsih, M.Si	3	0	VI
2	Analisis Data Uji Hidup	P	Nariza Wanti Wulan Sari, S.Si., M.Si	2	1	VI
Total				5	1	

Nama Dosen : Dr. M. Fathurahman, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Rancangan Percobaan	W	Memmi Nor Hayati, S.Si., M.Si	2	1	IV
2	Analisis Regresi	W	Dr. Darnah Andi Nohe, S.Si, M.Si	2	1	IV
3	Pengantar Metode Survei	W	Memmi Nor Hayati, S.Si., M.Si	2	1	IV
4	Teori Antrian	P	Dr. Sri Wahyuningsih, M.Si	3	0	VI
Total				9	3	

Nama Dosen : Dr. Sifriyani, S.Pd., M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Statistika Dasar	W	Ika Purnamasari, S.Si, M.Si	2	1	II
2	Pengantar Model Linier	W	Meiliyani Siringoringo, S.Si., M.Si	3	0	VI
3	Ekonometrika	P	Dr. Darnah Andi Nohe, S.Si, M.Si	2	1	VI
Total				7	2	

Nama Dosen : Ika Purnamasari, S.Si., M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Analisis Data Eksploratif	W	Dr. Darnah Andi Nohe, S.Si, M.Si	2	1	VI
2	Statistika Dasar	W	Dr. Sifriyani, S.Pd., M.Si	2	1	II
Total				4	2	

Nama Dosen : Memi Nor Hayati, S.Si., M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Pengantar Metode Survei	W	Dr. M. Fathurahman, S.Si., M.Si	2	1	IV
2	Rancangan Percobaan	W	Dr. M. Fathurahman, S.Si., M.Si	2	1	IV
3	Biostatistika	P	Dr. Darnah Andi Nohe, S.Si, M.Si	2	1	IV
Total				6	3	

Nama Dosen : Meiliyani Siringoringo, S.Si., M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Pengantar Model Linier	W	Dr. Sifriyani, S.Pd., M.Si	3	0	VI
2	Statistika Pengendalian Mutu	W	Dr. Sri Wahyuningsih, M.Si	2	1	IV
Total				5	1	

Nama Dosen : Surya Prangga, S.Si., M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Data Mining	W	Rito Goejantoro, S.Si., M.Si	2	1	IV
Total				2	1	

Ditetapkan di Samarinda

REKTOR,



Prof. H. Masjaya, M.Si.
NIP 196212311991031024

LAMPIRAN III
 KEPUTUSAN REKTOR UNIVERSITAS MULAWARMAN
 NOMOR 247 /UN17/HK/2022
 TANGGAL 12 JANUARI 2022
 TENTANG
 PENGANGKATAN DOSEN PENGAMPU MATA KULIAH
 PROGRAM MERDEKA BELAJAR KAMPUS MERDEKA
 (MBKM) SEMESTER GENAP TAHUN AKADEMIK
 2021/2022 PADA PROGRAM STUDI S1 STATISTIKA,
 S1 MATEMATIKA, S1 BIOLOGI, S1 KIMIA, S1 FISIKA
 DAN S1 GEOFISIKA FAKULTAS MATEMATIKA DAN
 ILMU PENGETAHUAN ALAM UNIVERSITAS
 MULAWARMAN.

PROGRAM STUDI : S1-MATEMATIKA

N O	KODE MK	MATA KULIAH	SKS/ SMT	JENIS MK (W/P)	KURIKULUM	DOSEN PENGAMPU MK
(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	210706604W005	Aljabar Linier Elementer	4/II	W	2021	Yuki Novia Nasution, S.Si., M.Sc Qonita Qurrota Ayun, S.Si., M.Sc
2	190706603W014	Kalkulus Peubah Banyak	3/IV	W	2019	Dr. Syaripuddin, M.Si Yuki Novia Nasution, S.Si., M.Sc
3	190706603W016	Fungsi Kompleks	3/IV	W	2019	Moh. Nurul Huda, S.Si., M.Si
4	190706603W017	Metode Numerik	3/IV	W	2019	Dr. Syaripuddin, M.Si Wasono, S.Si., M.Si
5	190706603W019	Persamaan Diferensial Biasa	3/IV	W	2019	Wasono, S.Si., M.Si Yuki Novia Nasution, S.Si., M.Sc
6	190706602P022	Teori Koding	2/IV	P	2019	Qonita Qurrota Ayun, S.Si., M.Sc
7	190706602P025	Teori Graf	2/IV	P	2019	Fidia Deny Tisna Amijaya, S.Si., M.Si Qonita Qurrota Ayun, S.Si., M.Sc
8	190706603W040	Pemodelan Matematika	3/VI	W	2019	Dr. Syaripuddin, M.Si Yuki Novia Nasution, S.Si., M.Sc
9	190706604P044	Persamaan Diferensial Numerik	4/VI	P	2019	Fidia Deny Tisna Amijaya, S.Si., M.Si Moh. Nurul Huda, S.Si., M.Si
10	190706603P050	Riset Operasi II	3/VI	P	2019	Dr. Syaripuddin, M.Si Wasono, S.Si., M.Si

Ditetapkan di Samarinda

REKTOR,



Prof. H. Masjaya, M.Si.
 NIP 196212311991031024

LAMPIRAN IV
KEPUTUSAN REKTOR UNIVERSITAS MULAWARMAN
NOMOR 547 /UN17/HK/2022
TANGGAL 12 JANUARI 2022
TENTANG

PENGANGKATAN DOSEN PENGAMPU MATA KULIAH
PROGRAM MERDEKA BELAJAR KAMPUS MERDEKA
(MBKM) SEMESTER GENAP TAHUN AKADEMIK
2021/2022 PADA PROGRAM STUDI S1 STATISTIKA,
S1 MATEMATIKA, S1 BIOLOGI, S1 KIMIA, S1 FISIKA
DAN S1 GEOFISIKA FAKULTAS MATEMATIKA DAN
ILMU PENGETAHUAN ALAM UNIVERSITAS
MULAWARMAN.

MATRIKS MATA KULIAH PROGRAM MERDEKA BELAJAR KAMPUS MERDEKA (MBKM)
SEMESTER GENAP TAHUN AKADEMIK 2021/2022 PROGRAM STUDI S1 MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM UNIVERSITAS MULAWARMAN

Nama Dosen : Dr. Syaripuddin, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Kalkulus Peubah Banyak	W	Yuki Novia Nasution, S.Si., M.Sc	3	0	IV
2	Metode Numerik	W	Wasono, S.Si., M.Si	2	1	IV
3	Pemodelan Matematika	W	Yuki Novia Nasution, S.Si., M.Sc	3	0	VI
4	Riset Operasi II	P	Wasono, S.Si., M.Si	3	0	VI
Total				11	1	

Nama Dosen : Yuki Novia Nasution, S.Si., M.Sc

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Aljabar Linier Elementer	W	Qonita Qurrota A'yun, S.Si., M.Sc	4	0	II
2	Kalkulus Peubah Banyak	W	Dr. Syaripuddin, M.Si	3	0	IV
3	Persamaan Diferensial Biasa	W	Wasono, S.Si., M.Si	3	0	IV
4	Pemodelan Matematika	W	Dr. Syaripuddin, M.Si	3	0	VI
Total				13	0	

Nama Dosen : Fidia Deny Tisna Amijaya, S.Si., M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Teori Graf	P	Qonita Qurrota A'yun, S.Si., M.Sc	2	0	IV
2	Persamaan Diferensial Numerik	P	Moh. Nurul Huda, S.Si., M.Si	3	1	VI
Total				5	1	

Nama Dosen : Qonita Qurrota A'yun, S.Si., M.Sc

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Aljabar Linier Elementer	W	Yuki Novia Nasution, S.Si., M.Sc	4	0	II
2	Teori Koding	P	-	2	0	IV
3	Teori Graf	P	Fidia Deny Tisna Amijaya, S.Si., M.Si	2	0	IV
Total				8	0	

Nama Dosen : Wasono, S.Si., M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Metode Numerik	W	Dr. Syaripuddin, M.Si	2	1	IV
2	Persamaan Diferensial Biasa	W	Yuki Novia Nasution, S.Si., M.Sc	3	0	IV
3	Riset Operasi II	P	Dr. Syaripuddin, M.Si	3	0	VI
Total				8	1	

Nama Dosen : Moh. Nurul Huda, S.Si., M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Fungsi Kompleks	W	-	3	0	IV
2	Persamaan Differensial Numerik	P	Fidia Deny Tisna Amijaya. S.Si., M.Si	3	1	VI
Total				6	1	

Ditetapkan di Samarinda

REKTOR,



H. Masjaya, M.Si.

NIP 196212311991031024

LAMPIRAN V

KEPUTUSAN REKTOR UNIVERSITAS MULAWARMAN

NOMOR 547 /UN17/HK/2022

TANGGAL 12 JANUARI 2022

TENTANG

PENGANGKATAN DOSEN PENGAMPU MATA KULIAH PROGRAM MERDEKA BELAJAR KAMPUS MERDEKA (MBKM) SEMESTER GENAP TAHUN AKADEMIK 2021/2022 PADA PROGRAM STUDI S1 STATISTIKA, S1 MATEMATIKA, S1 BIOLOGI, S1 KIMIA, S1 FISIKA DAN S1 GEOFISIKA FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM UNIVERSITAS MULAWARMAN.

PROGRAM STUDI : S1-BIOLOGI

N O	KODE MK	MATA KULIAH	SKS/ SMT/ KLS	JENIS MK (W/P)	KURIKULUM	DOSEN PENGAMPU MK
(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	210702603W003	Anatomi Hewan	3/II	W	2021	Dr. Retno Aryani, M.Si Rudy Agung Nugroho, M.Si., Ph.D
2	190702603W022	Fisiologi Tumbuhan	3/IV/ A&B	W	2019	Dr. Dwi Susanto, M.Si Dr. Hetty Manurung, M.Si
3	190702603P055	Mikrobiologi Lingkungan	3/VI	P	2019	Imam Rosadi, S.Si., M.Si Dr. Ir. Samsurianto, M.Si
4	190702603P057	Genetika Molekuler Mikrobia	3/VI	P	2019	Dr. rer. nat. Bodhi Dharma, M.Si Imam Rosadi, S.Si., M.Si
5	190702603P061	Endrokrinologi	3/VI	P	2019	Dr. Retno Aryani, M.Si Rudy Agung Nugroho, M.Si., Ph.D

Ditetapkan di Samarinda

REKTOR,



H. Masjaya, M.Si.

NIP. 196212311991031024

LAMPIRAN VI
KEPUTUSAN REKTOR UNIVERSITAS MULAWARMAN
NOMOR 547 /UN17/HK/2022
TANGGAL 12 JANUARI 2022
TENTANG
PENGANGKATAN DOSEN PENGAMPU MATA KULIAH
PROGRAM MERDEKA BELAJAR KAMPUS MERDEKA
(MBKM) SEMESTER GENAP TAHUN AKADEMIK
2021/2022 PADA PROGRAM STUDI S1 STATISTIKA,
S1 MATEMATIKA, S1 BIOLOGI, S1 KIMIA, S1 FISIKA
DAN S1 GEOFISIKA FAKULTAS MATEMATIKA DAN
ILMU PENGETAHUAN ALAM UNIVERSITAS
MULAWARMAN.

MATRIKS MATA KULIAH PROGRAM MERDEKA BELAJAR KAMPUS MERDEKA (MBKM)
SEMESTER GENAP TAHUN AKADEMIK 2021/2022 PROGRAM STUDI S1 BIOLOGI
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS MULAWARMAN

Nama Dosen : Imam Rosadi, S.Si., M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Mikrobiologi Lingkungan	P	Dr. Syafrizal, MP	2	1	VI
2	Genetika Molekuler Mikrobial	P	Dr. rer. nat. Bodhi Dharma, M.Si	2	1	VI
Total				4	2	

Nama Dosen : Dr. Ir. Samsurianto, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Mikrobiologi Lingkungan	P	Imam Rosadi, S.Si., M.Si	2	1	VI
Total				2	1	

Nama Dosen : Dr. rer. nat. Bodhi Dharma, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Genetika Molekuler Mikrobial	P	Imam Rosadi, S.Si., M.Si	2	1	VI
Total				2	1	

Nama Dosen : Dr. Dwi Susanto, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Fisiologi Tumbuhan	W	Dr. Hetty Manurung, M.Si	2	1	IV
Total				2	1	

Nama Dosen : Dr. Hetty Manurung, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Fisiologi Tumbuhan	W	Dr. Dwi Susanto, M.Si	2	1	IV
Total				2	1	

Nama Dosen : Rudy Agung Nugroho, M.Si., Ph.D

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Anatomi Hewan	W	Dr. Retno Aryani, M.Si	2	1	II
2	Endokrinologi	P	Dr. Retno Aryani, M.Si	2	1	VI
Total				4	2	

Nama Dosen : Dr. Retno Aryani, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Anatomi Hewan	W	Rudy Agung Nugroho, M.Si., Ph.D	2	1	II
2	Endokrinologi	P	Rudy Agung Nugroho, M.Si., Ph.D	2	1	VI
Total				4	2	

Ditetapkan di Samarinda



REKTOR,

H. Masjaya, M.Si.

NIP. 196212311991031024

LAMPIRAN VII
 KEPUTUSAN REKTOR UNIVERSITAS MULAWARMAN
 NOMOR 547 /UN17/HK/2022
 TANGGAL 12 JANUARI 2022
 TENTANG
 PENGANGKATAN DOSEN PENGAMPU MATA KULIAH
 PROGRAM MERDEKA BELAJAR KAMPUS MERDEKA
 (MBKM) SEMESTER GENAP TAHUN AKADEMIK
 2021/2022 PADA PROGRAM STUDI S1 STATISTIKA,
 S1 MATEMATIKA, S1 BIOLOGI, S1 KIMIA, S1 FISIKA
 DAN S1 GEOFISIKA FAKULTAS MATEMATIKA DAN
 ILMU PENGETAHUAN ALAM UNIVERSITAS
 MULAWARMAN.

PROGRAM STUDI : S1-KIMIA

N O	KODE MK	MATA KULIAH	SKS/ SMT	JENIS MK (W/P)	KURIKULUM	DOSEN PENGAMPU MK
(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	190703602P033	Ikatan Kimia	2/IV	P	2019	Dr. Rahmat Gunawan, M.Si Veliyana Londong, S.Si., M.Si
2	190703602P038	Kimia Batu Bara	2/IV	P	2019	Dr. Saibun Sitorus, M.Si Prof. Dr. Ir. Daniel, M.Si
3	190703603W024	Sintesa Kimia Anorganik	3/VI	W	2019	Dr. Noor Hindryawati, M.Si Irfan Ashari Hiyahara, M.Si
4	190703602P061	Nanoteknologi dan Nanomaterial	2/VI	P	2019	Dr. Noor Hindryawati, M.Si Irfan Ashari Hiyahara, M.Si
5	190703602P062	Analisis Runut	2/VI	P	2019	Prof. Dr. Aman Sentosa Panggabean, M.Si Drs. H. Alimuddin, M.Si
6	190703602P069	Oleokimia Dasar	2/VI	P	2019	Dr. Chairul Saleh, M.Si Ritson Purba, S.Si. M.Si
7	190703602P070	Biokimia Medisinal	2/VI	P	2019	Ritbey Ruga, M.P., Ph.D Djihan Ryn Pratiwi, S.Si., M.Si

Ditetapkan di Samarinda

REKTOR,



Prof. Dr. H. Masjaya, M.Si.
 NIP 196212311991031024

LAMPIRAN VIII
KEPUTUSAN REKTOR UNIVERSITAS MULAWARMAN
NOMOR 547 /UN17/HK/2022
TANGGAL 12 JANUARI 2022
TENTANG
Matriks Mata Kuliah Program Merdeka Belajar Kampus Merdeka (MBKM) Semester Genap Tahun Akademik 2021/2022 Program Studi S1 Kimia Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Mulawarman.

Matriks Mata Kuliah Program Merdeka Belajar Kampus Merdeka (MBKM)
Semester Genap Tahun Akademik 2021/2022 Program Studi S1 Kimia
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Mulawarman

Nama Dosen : Prof. Dr. Ir. Daniel, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Kimia Batu Bara	P	Dr. Saibun Sitorus, M.Si	2	0	IV
Total				2	0	

Nama Dosen : Dr. Saibun Sitorus, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Kimia Batu Bara	P	Prof. Dr. Ir. Daniel, M.Si	2	0	IV
Total				2	0	

Nama Dosen : Prof. Dr. Aman Sentosa Panggabean, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Analisis Runut	P	Drs. H. Alimuddin, M.Si	2	0	VI
Total				2	0	

Nama Dosen : Dr. H. Alimuddin, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Analisis Runut	P	Prof. Dr. Aman Sentosa Panggabean, M.Si	2	0	VI
Total				2	0	

Nama Dosen : Dr. Chairul Saleh, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Oleokimia Dasar	P	Ritson Purba, S.Si. M.Si	2	0	VI
Total				2	0	

Nama Dosen : Ritson Purba, S.Si., M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Oleokimia Dasar	P	Dr. Chairul Saleh, M.Si	2	0	VI
Total				2	0	

Nama Dosen : Dr. Noor Hindryawati, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Sintesa Kimia Anorganik	W	Irfan Ashari Hiyahara, M.Si	3	0	VI
2	Nanoteknologi dan Nanomaterial	P	Irfan Ashari Hiyahara, M.Si	2	0	VI
Total				5	0	

Nama Dosen : Irfan Ashari Hiyahara, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Sintesa Kimia Anorganik	W	Dr. Noor Hindryawati, M.Si	3	0	VI
2	Nanoteknologi dan Nanomaterial	P	Dr. Noor Hindryawati, M.Si	2	0	VI
Total				5	0	

Nama Dosen : Dr. Rahmat Gunawan, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Ikatan Kimia	P	Veliyana Londong, S.Si., M.Si	2	0	IV
Total				2	0	

Nama Dosen : Veliyana Londong Allo, S.Si., M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Ikatan Kimia	P	Dr. Rahmat Gunawan, M.Si	2	0	IV
Total				2	0	

Nama Dosen : Ritbey Ruga, MP., Ph.D

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Biokimia Medisinal	P	Djihan Ryn Pratiwi, S.Si., M.Si	2	0	VI
Total				2	0	

Nama Dosen : Djihan Ryn Pratiwi, S.Si., M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Biokimia Medisinal	P	Ritbey Ruga, M.P., Ph.D	2	0	VI
Total				2	0	

Ditetapkan di Samarinda

REKTOR,



H. Masjaya, M.Si.
NIP 196212311991031024

LAMPIRAN IX

KEPUTUSAN REKTOR UNIVERSITAS MULAWARMAN

NOMOR 547 /UN17/HK/2022

TANGGAL 12 JANUARI 2022

TENTANG

PENGANGKATAN DOSEN PENGAMPU MATA KULIAH PROGRAM MERDEKA BELAJAR KAMPUS MERDEKA (MBKM) SEMESTER GENAP TAHUN AKADEMIK 2021/2022 PADA PROGRAM STUDI S1 STATISTIKA, S1 MATEMATIKA, S1 BIOLOGI, S1 KIMIA, S1 FISIKA DAN S1 GEOFISIKA FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM UNIVERSITAS MULAWARMAN.

PROGRAM STUDI : S1-FISIKA

N O	KODE MK	MATA KULIAH	SKS/ SMT	JENIS MK (W/P)	KURIKULUM	DOSEN PENGAMPU MK
(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	190704602W008	Fisika Lingkungan	2/II	W	2019	Dr. Mislan, M.Si Dadan Hamdani, M.Si Kadek Subagiada, M.Si
2	190704603P049	Biofisika	3/IV	P	2019	Devina Rayzy Perwitasari Sutaji Putri, S.Si., M.Sc
3	190704602W037	Kapita Selekta	2/VI	W	2019	Wahidah, S.Si., MT Rahmiati, S.Si., M.Sc
4	190704603P070	Pemodelan Oseanografi	3/VI	P	2019	Dr. Eng. Idris Mandang, M.Si Dr. Sc. Mustaid Yusuf, M.Si
5	190704603P066	Pengantar Mikro Prosesor	3/VI	P	2019	Dr. Syahrir, M.Si Ahmad Zarkasi, S.Si., M.Si

Ditetapkan di Samarinda

REKTOR,



H. Masjaya, M.Si.

NIP 196212311991031024

LAMPIRAN X
 KEPUTUSAN REKTOR UNIVERSITAS MULAWARMAN
 NOMOR 547 /UN17/HK/2022
 TANGGAL 12 JANUARI 2022
 TENTANG
 MATRIKS MATA KULIAH PROGRAM MERDEKA
 BELAJAR KAMPUS MERDEKA (MBKM) SEMESTER
 GENAP TAHUN AKADEMIK 2021/2022 PROGRAM
 STUDI S1 FISIKA FAKULTAS MATEMATIKA DAN ILMU
 PENGETAHUAN ALAM UNIVERSITAS MULAWARMAN.

MATRIKS MATA KULIAH PROGRAM MERDEKA BELAJAR KAMPUS MERDEKA (MBKM)
 SEMESTER GENAP TAHUN AKADEMIK 2021/2022 PROGRAM STUDI S1 FISIKA
 FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
 UNIVERSITAS MULAWARMAN

Nama Dosen : Dr. Mislan, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Fisika Lingkungan	W	Dadan Hamdani, M.Si	2	0	II
Total				2	0	

Nama Dosen : Dadan Hamdani, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Fisika Lingkungan	W	Dr. Mislan, M.Si	2	0	II
Total				2	0	

Nama Dosen : Dr. Syahrir, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Pengantar MikroProsesor	P	Ahmad Zarkasi, S.Si., M.Si	2	1	VI
Total				2	1	

Nama Dosen : Ahmad Zarkasi, S.Si., M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Pengantar Mikroprosesor	P	Dr. Syahrir, M.Si	2	1	VI
Total				2	1	

Nama Dosen : Dr. Eng. Idris Mandang, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Pemodelan Oseanografi	P	Dr. Sc. Mustaid Yusuf, M.Si	2	1	VI
Total				2	1	

Nama Dosen : Dr. Sc. Mustaid Yusuf, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Pemodelan Oseanografi	P	Dr. Eng. Idris Mandang, M.Si	2	1	VI
Total				2	1	

Nama Dosen : Kadek Subagiada, S.Si., M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Biofisika	P	Devina Razy Perwitasari Sutaji Putri, S.Si., M.Sc	3	0	VI
Total				3	0	

Nama Dosen : Devina Rayzy Perwitasari Sutaji Putri, S.Si., M.Sc

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Biofisika	P	Kadek Subagiada, S.Si., M.Si	3	0	IV
Total				3	0	

Nama Dosen : Wahidah, S.Si., MT

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Kapita Selekt	W	Rahmiati, S.Si., M.Sc	2	0	VI
Total				2	0	

Nama Dosen : Rahmiati, S.Si., M.Sc

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Kapita Selekt	W	Wahidah, S.Si., MT	2	0	VI
Total				2	0	

Ditetapkan di Samarinda



REKTOR,

H. Masjaya, M.Si.

NIP 196212311991031024

LAMPIRAN XI
KEPUTUSAN REKTOR UNIVERSITAS MULAWARMAN
NOMOR 547 /UN17/HK/2022
TANGGAL 12 JANUARI 2022
TENTANG
PENGANGKATAN DOSEN PENGAMPU MATA KULIAH
PROGRAM MERDEKA BELAJAR KAMPUS MERDEKA
(MBKM) SEMESTER GENAP TAHUN AKADEMIK
2021/2022 PADA PROGRAM STUDI S1 STATISTIKA,
S1 MATEMATIKA, S1 BIOLOGI, S1 KIMIA, S1 FISIKA
DAN S1 GEOFISIKA FAKULTAS MATEMATIKA DAN
ILMU PENGETAHUAN ALAM UNIVERSITAS
MULAWARMAN.

PROGRAM STUDI : S1-GEOFISIKA

NO	KODE MK	MATA KULIAH	SKS/ SMT	JENIS MK (W/P)	KURIKULUM	DOSEN PENGAMPU MK
(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	190707602P043	Geologi Cekungan Kutai	2/IV	P	2019	Dr. Syahrir, M.Si Dr. Ery Arifullah, Ph.D
2	190707603W030	Metode Geolistrik dan Elektromagnetik	3/VI	W	2019	Dr. Djayus, M.T Dr. Syahrir, M.Si
3	190707602W032	Kapita Selekta	2/VI	W	2019	Wahidah, S.Si, M.T Rahmiati, S.Si., M.Sc
4	190707603W035	Kuliah Lapangan Geofisika	3/VI	P	2019	Dr. Supriyanto, M.T Dr. Djayus, M.T
5	190707603P065	Pemodelan Oseanografi	3/VI	P	2019	Dr. Eng. Idris Mandang, M.Si Dr. Sc. Mustaid Yusuf, M.Si

Ditetapkan di Samarinda



REKTOR,

Prof. Dr. H. Masjaya, M.Si.

REKTOR 196212311991031024

LAMPIRAN XII

KEPUTUSAN REKTOR UNIVERSITAS MULAWARMAN

NOMOR 847 /UN17/HK/2022

TANGGAL 12 JANUARI 2022

TENTANG

MATRIKS MATA KULIAH PROGRAM MERDEKA
BELAJAR KAMPUS MERDEKA (MBKM) SEMESTER
GENAP TAHUN AKADEMIK 2021/2022 PROGRAM
STUDI S1 GEOFISIKA FAKULTAS MATEMATIKA DAN
ILMU PENGETAHUAN ALAM UNIVERSITAS
MULAWARMAN.

MATRIKS MATA KULIAH PROGRAM MERDEKA BELAJAR KAMPUS MERDEKA (MBKM)
SEMESTER GENAP TAHUN AKADEMIK 2021/2022 PROGRAM STUDI S1 GEOFISIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS MULAWARMAN

Nama Dosen : Dr. Eng. Idris Mandang, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Pemodelan Oseanografi	P	Dr. Sc. Mustaid Yusuf, M.Si	2	1	VI
Total				2	1	

Nama Dosen : Dr. Sc. Mustaid Yusuf, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Pemodelan Oseanografi	P	Dr. Eng. Idris Mandang, M.Si	2	1	VI
Total				2	1	

Nama Dosen : Dr. Syahrir, M.Si

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Geologi Cekungan Kutai	P	Dr. Ery Arifullah, Ph.D	2	0	IV
2	Metode Geolistrik dan Elektromagnetik	W	Dr. Djayus, MT	2	1	VI
Total				4	1	

Nama Dosen : Dr. Djayus, MT

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Metode Geolistrik dan Elektromagnetik	W	Dr. Syahrir, M.Si	2	1	VI
2	Kuliah Lapangan Geofisika	P	Dr. Supriyanto, M.T	0	3	VI
Total				2	4	

Nama Dosen : Dr. Supriyanto, MT

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Kuliah Lapangan Geofisika	P	Dr. Djayus, MT	0	3	VI
Total				0	3	

Nama Dosen : Wahidah, S.Si., MT

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Kapita Selekt	W	Rahmiati, S.Si., M.Sc	2	0	VI
Total				2	0	

Nama Dosen : Rahmiati, S.Si., M.Sc

NO	MATA KULIAH	W/P	DOSEN PARTNER	SKS		SMT
				TEORI	PRAKTEK	
1	Kapita Selekta	W	Wahidah, S.Si., M.T	2	0	VI
Total				2	0	

Ditetapkan di Samarinda



REKTOR,

H. Masjaya, M.Si.
NIP. 6212311991031024

PENGANTAR MIKROKONTROLER

Pertemuan 1

Ahmad Zarkasi



KULIAH PENDAHULUAN

**PENGANTAR TEKNOLOGI
MIKROKONTROLER**

Capaian Pembelajaran

- Mahasiswa memahami perbedaan mikrokontroler dan mikroprosesor
- Mahasiswa memahami perkembangan teknologi mikrokontroler
- Mahasiswa mengetahui implementasi mikrokontroler

Apakah Mikrokontroler itu?

Apakah sama dengan Mikroprosesor?

Apa bedanya dengan Komputer?

Computer is a machine that can be instructed to carry out sequences of arithmetic or logical operations automatically via computer programming.

(Source: wikipedia.org).

Microprocessor is an electronic component that is used by a computer to do its work. It is a central processing unit on a single integrated circuit.

(Source: wikipedia.org).

Computer Vs Microcontroller

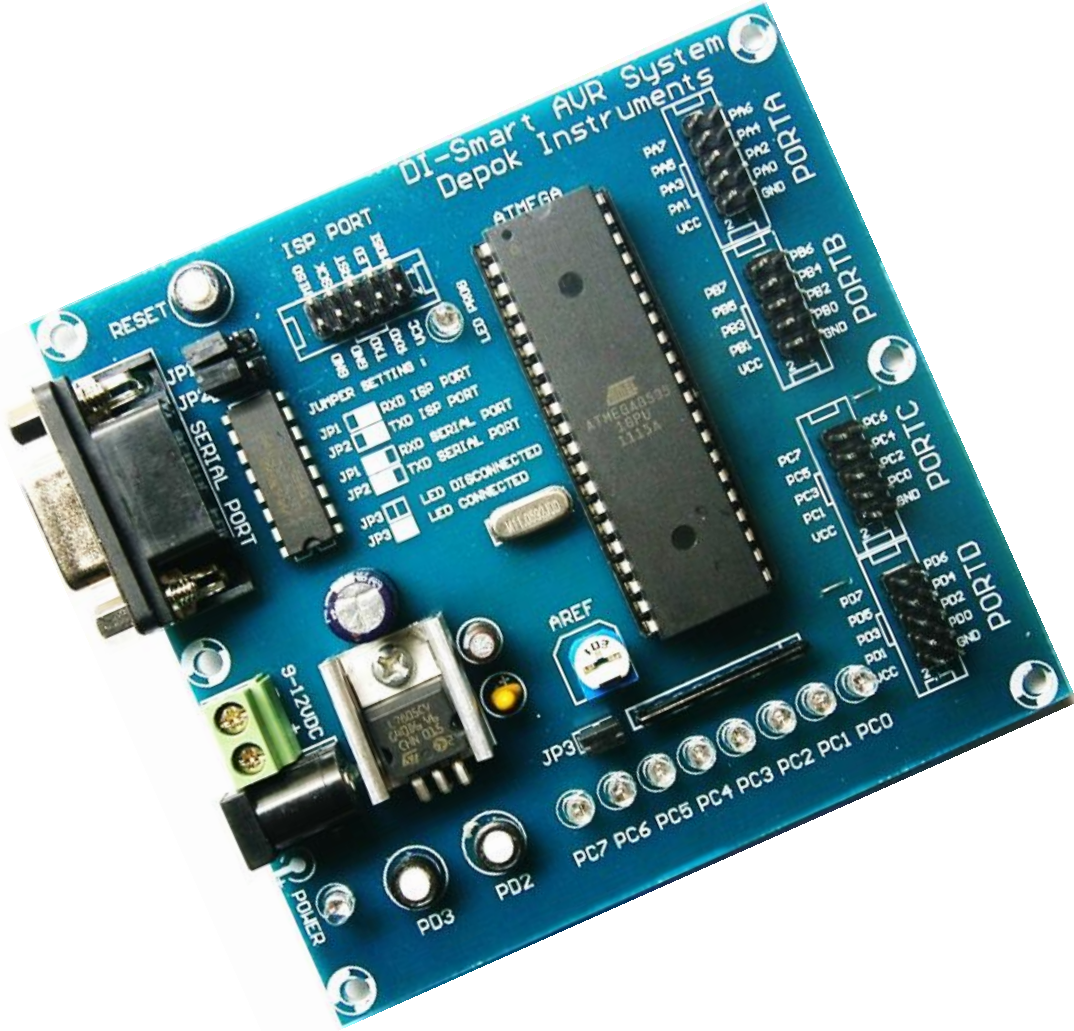


- ❖ CPU
- ❖ HDD/SSD & RAM
- ❖ System Clock
- ❖ Peripheral
- ❖ Menjalankan berbagai fungsi dan dapat dieksekusi secara bersamaan



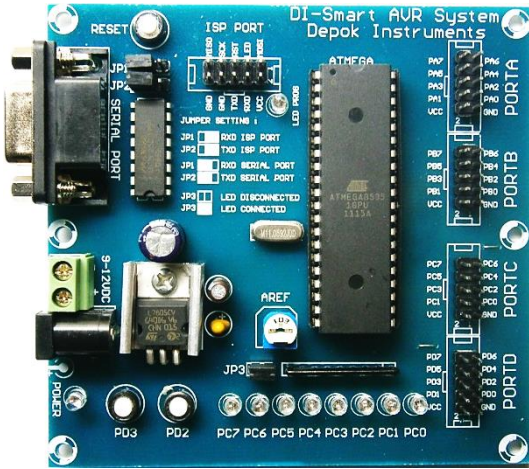
- ❖ CPU
- ❖ Memory
- ❖ System Clock
- ❖ Peripheral
- ❖ Dibuat untuk menjalankan fungsi khusus

Apakah Mikrokontroler itu?



A **microcontroller** is a small and low-cost microcomputer, which is designed to perform the specific tasks of embedded systems (on chip/IC).

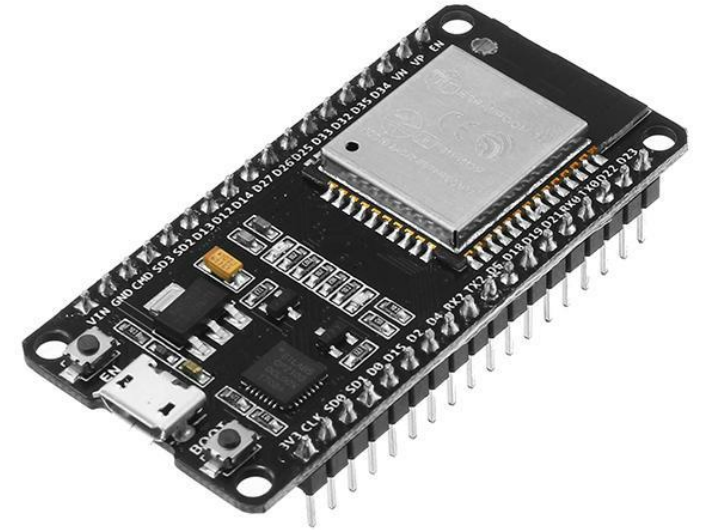
Beberapa jenis modul mikrokontroler yang populer



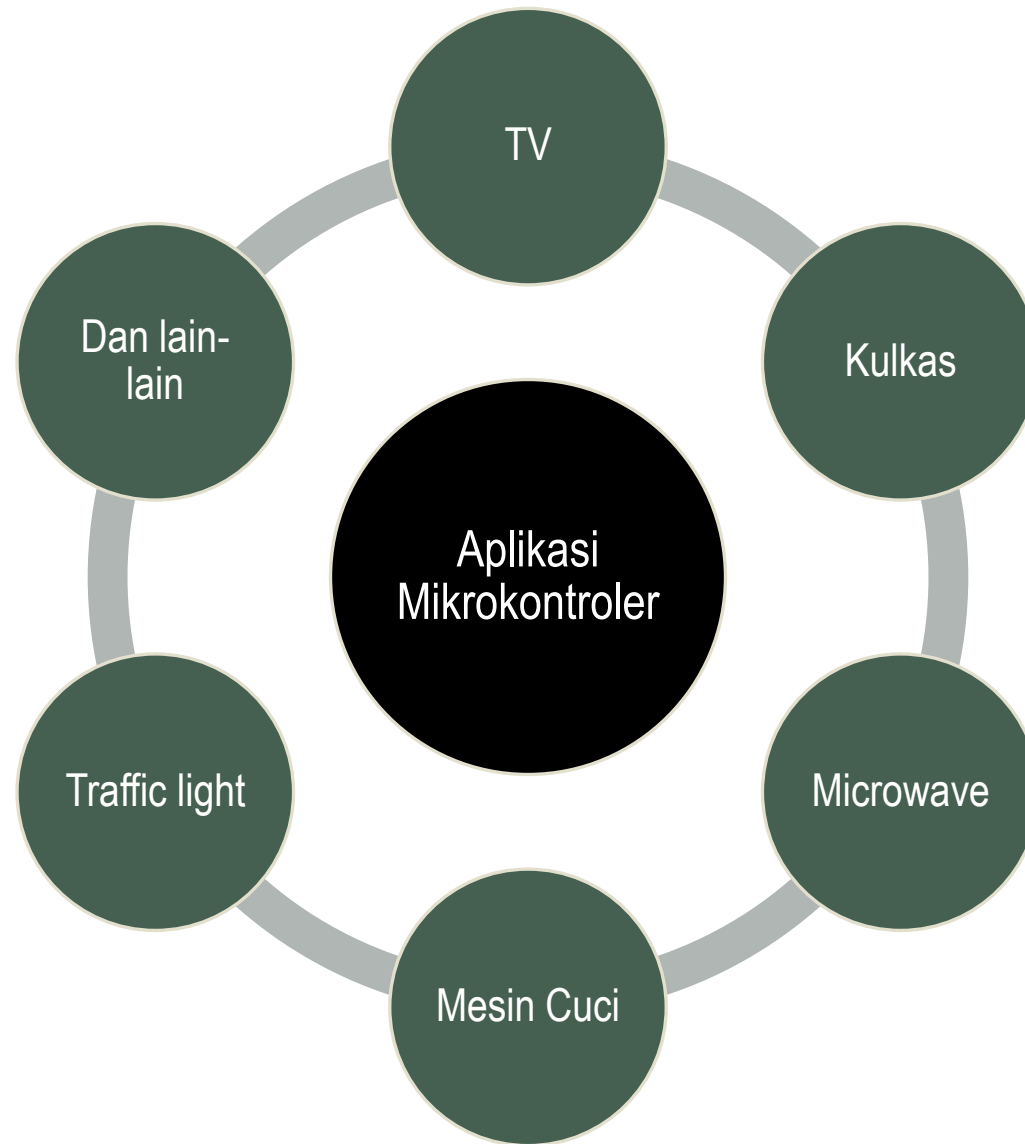
AVR



Arduino



ESP32



Microcontroller Vs Microprocessor

Microcontroller	Microprocessor
Microcontrollers are used to execute a single task within an application.	Microprocessors are used for big applications.
Its designing and hardware cost is low.	Its designing and hardware cost is high.
Easy to replace.	Not so easy to replace.
It is built with CMOS technology, which requires less power to operate.	Its power consumption is high because it has to control the entire system.
It consists of CPU, RAM, ROM, I/O ports.	It doesn't consist of RAM, ROM, I/O ports. It uses its pins to interface to peripheral devices.

(Source: tutorialspoint.com)

MIKROKONTROLER 8051

8051 dan Turunannya

Merupakan jenis mikrokontroler 8bit yang paling terkenal, dikeluarkan oleh INTEL Corporation pada tahun 1981. Keberadaannya sudah sangat lama dan turunannya sangat banyak (ratusan ribu dari berbagai produsen).

Memory Program Internal

- Menggunakan *on-chip ROM*
- Ukuran *on-chip ROM* : 0kByte(8031), 4kByte(8051), 8kByte(8052)

RAM Internal

- Ukuran : 128Byte(8051), 256Byte(8052)

Memory Eksternal

- Memory program dan data dapat dikembangkan sampai 64kByte menggunakan memory eksternal.

MIKROKONTROLER 8051 (Lanjutan)

ATMEL

- Pertama kali mengenalkan AT89C51 dengan *programmable flash memory* (bisa dihapus dan ditulis kembali)
- Juga mengenalkan AT89S5X dengan kemampuan *In System Programmable*

INTEL

- Pendesain awal 8051
- Dokumentasi yang terkait dengan 8051 bisa dilihat di :
http://developer.intel.com/design/mcs51/docs_mcs51.htm.

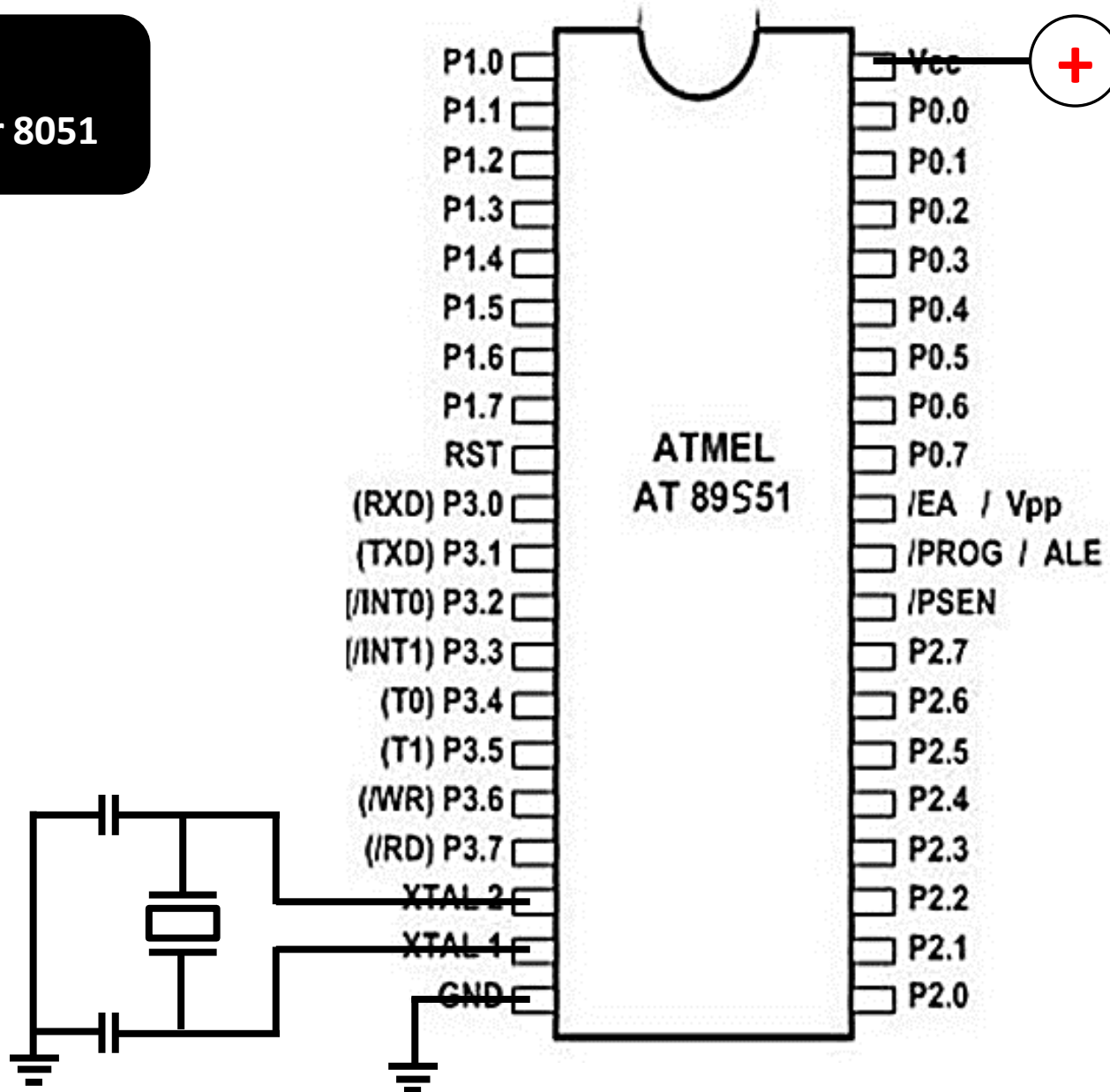
MAXIM (DALLAS SEMICONDUCTOR)

- Produk dari Maxim bisa dilihat di :
http://www.maxim-ic.com/products/microcontrollers/high_speed.cfm

PHILLIPS

- Produk dari Phillips bisa dilihat di :
<http://www.standardics.philips.com/products/microcontrollers/>

Contoh
Mikrokontroler 8051



MIKROKONTROLER AVR

AVR (dari ATMEL)

Merupakan mikrokontroler 8-bit (*RISC*, didasari dari *Harvard architecture*), dengan instruksi selebar 16bit (8 bit opcode). Mikrokontroler ini merupakan pesaing PIC.

Memory Program

- Menggunakan *flash memory* (sampai 256k)
- Ukuran *flash memory* kadang bisa dilihat dari nama IC-nya (ATmega64x :64 kByte)
- AVR dapat mengambil instruksi berikutnya sementara instruksi saat ini masih dikerjakan (teknik *single level pipeline*)
- Sebagian besar instruksi butuh hanya 1 atau 2 siklus instruksi. Oleh karena itu AVR merupakan mikrokontroler yang relatif lebih cepat bila dibandingkan mikrokontroler 8bit lainnya.

Memory Data Internal

- Terbagi sbb : register internal (sebanyak 32 register 8bit), register I/O (sebanyak 64) dan SRAM (sampai 8k)

EEPROM Internal

- Sebagian AVR menyediakan EEPROM internal untuk penyimpanan data semi permanen (sampai 4kB).

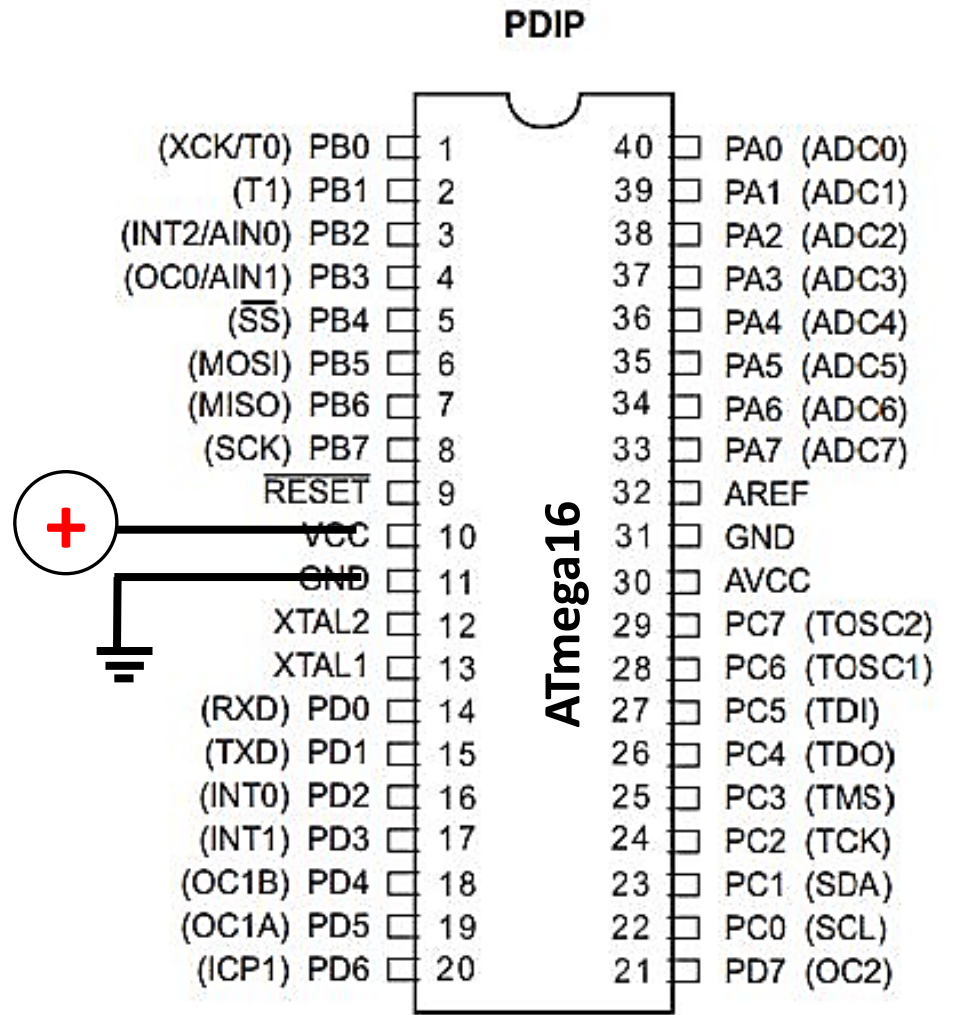
Kecepatan MCU

- Menggunakan clock sampai 16 MHz, sebagian sampai 20MHz

Pin Descriptions

- ☐ **VCC (pin number 10)**
Digital supply voltage
- ☐ **GND (pin number 11 & 31)**
Ground
- ☐ **PORT A (PA7..PA0/ pin number 33..40)**
General I/O, Analog input for A/D Converter
- ☐ **PORT B (PB0..PB7/ pin number 22..29)**
General I/O, Timer/Counter, Ext. Interrupt, Analog Comparator, SPI
- ☐ **PORT C (PC0..PC7/ pin number 22..29)**
General I/O, TWI, JTAG, Timer oscillator
- ☐ **PORT D (PD0..PD7/ pin number 14..21)**
General I/O, USART, Ext. Interrupt, Timer/Counter
- ☐ **RESET (pin number 9)**
Reset MCU
- ☐ **XTAL1 & XTAL2 (pin number 12 & 13)**
Input and output of inverting oscillator
- ☐ **AVCC (pin number 30)**
Supply voltage pin for Port A and the A/D Converter
- ☐ **AREFF (pin number 32)**
Analog reference for A/D Converter

Contoh Mikrokontroler AVR



MIKROKONTROLER AVR (Lanjutan)

KELUARGA AVR

tinyAVR

- Memori program : 1-8 kB
- IC : 8-20 pin
- Periferal dalam tinyAVR terbatas

megaAVR

- Memori program : 4-256 kB
- IC : 28-100 pin
- Mendukung lebih banyak kumpulan instruksi (perkalian dll)
- Menyediakan periferal yang lebih lengkap

AVR untuk aplikasi tertentu

- Contoh untuk *LCD controller, USB controller, advanced PWM* etc.

CATATAN

- Sangat cocok bila kita membutuhkan MCU yang cukup cepat.
- Memudahkan kita yang ingin menggunakan kontroler seperti : CAN, USB, Ethernet dll, karena telah terintegrasi dalam sebagian periferal.

SEKIAN DAN TERIMA KASIH

PENGANTAR MIKROKONTROLER

Pertemuan 2

Ahmad Zarkasi

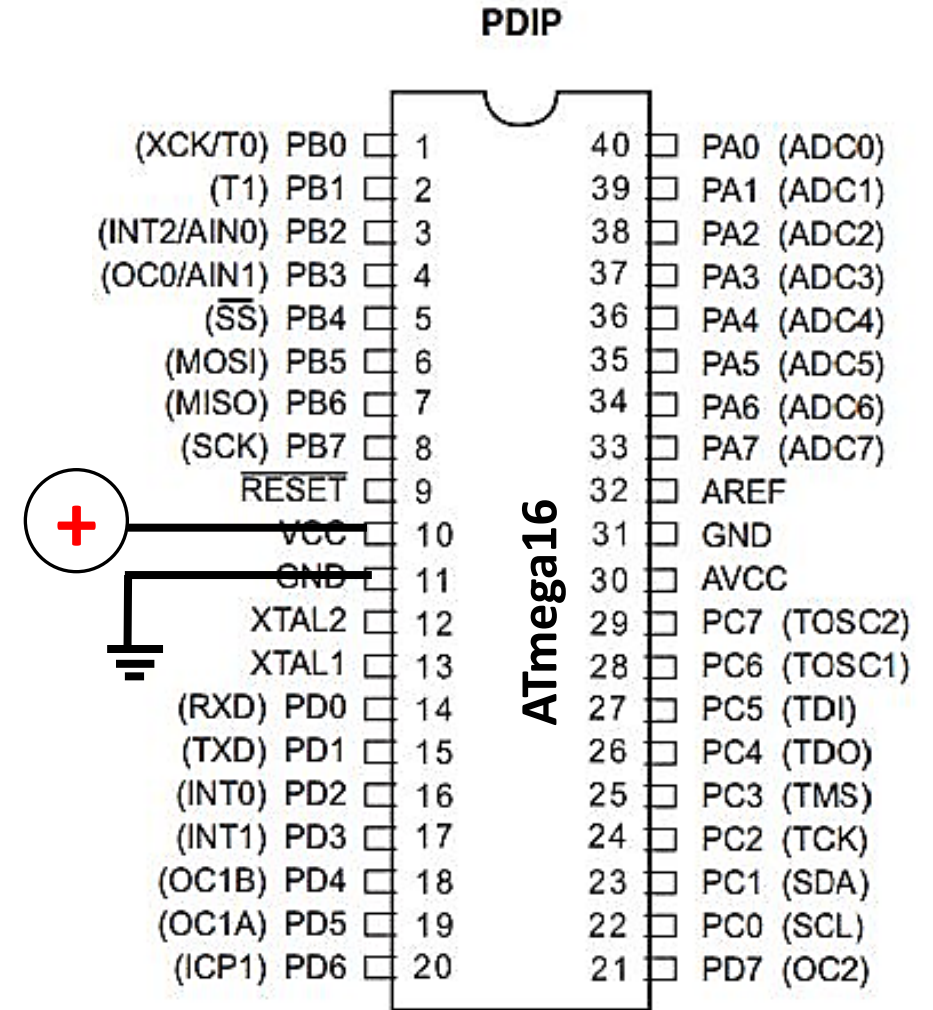


MATERI BAHASAN

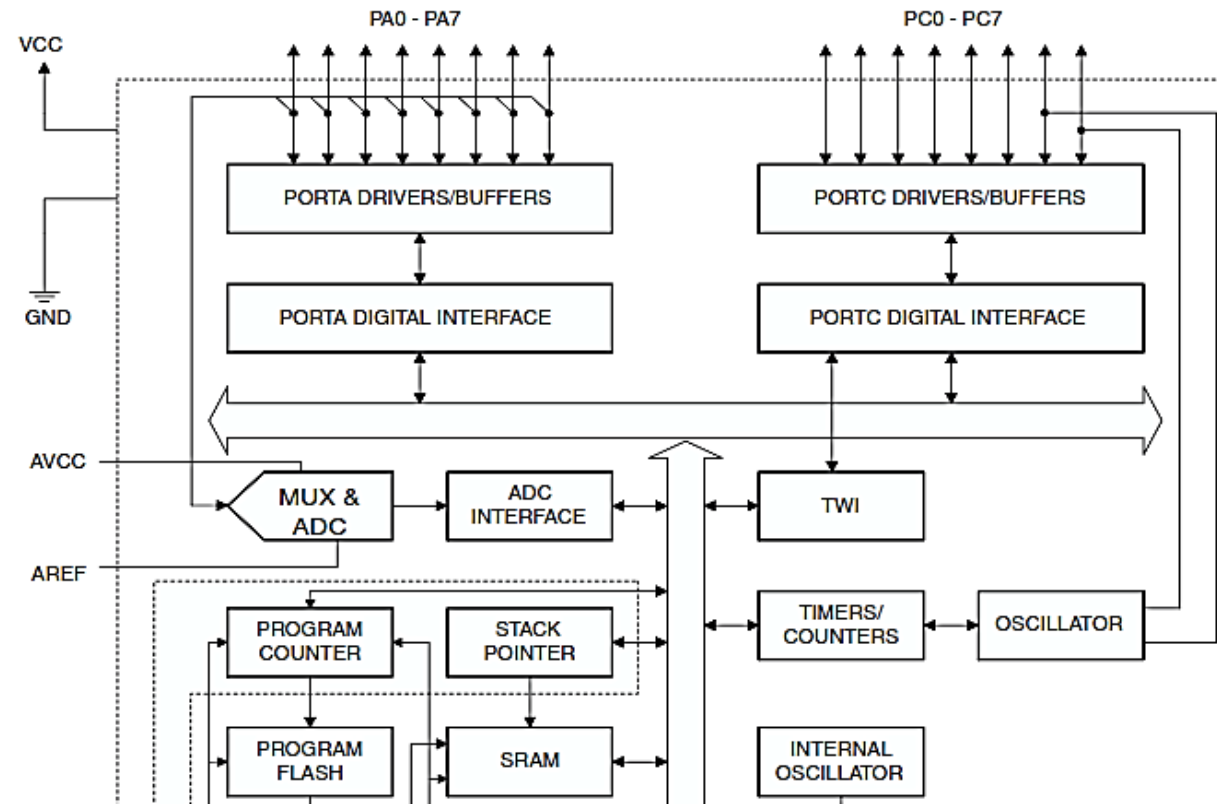
ARSITEKTUR AVR

Pin Descriptions

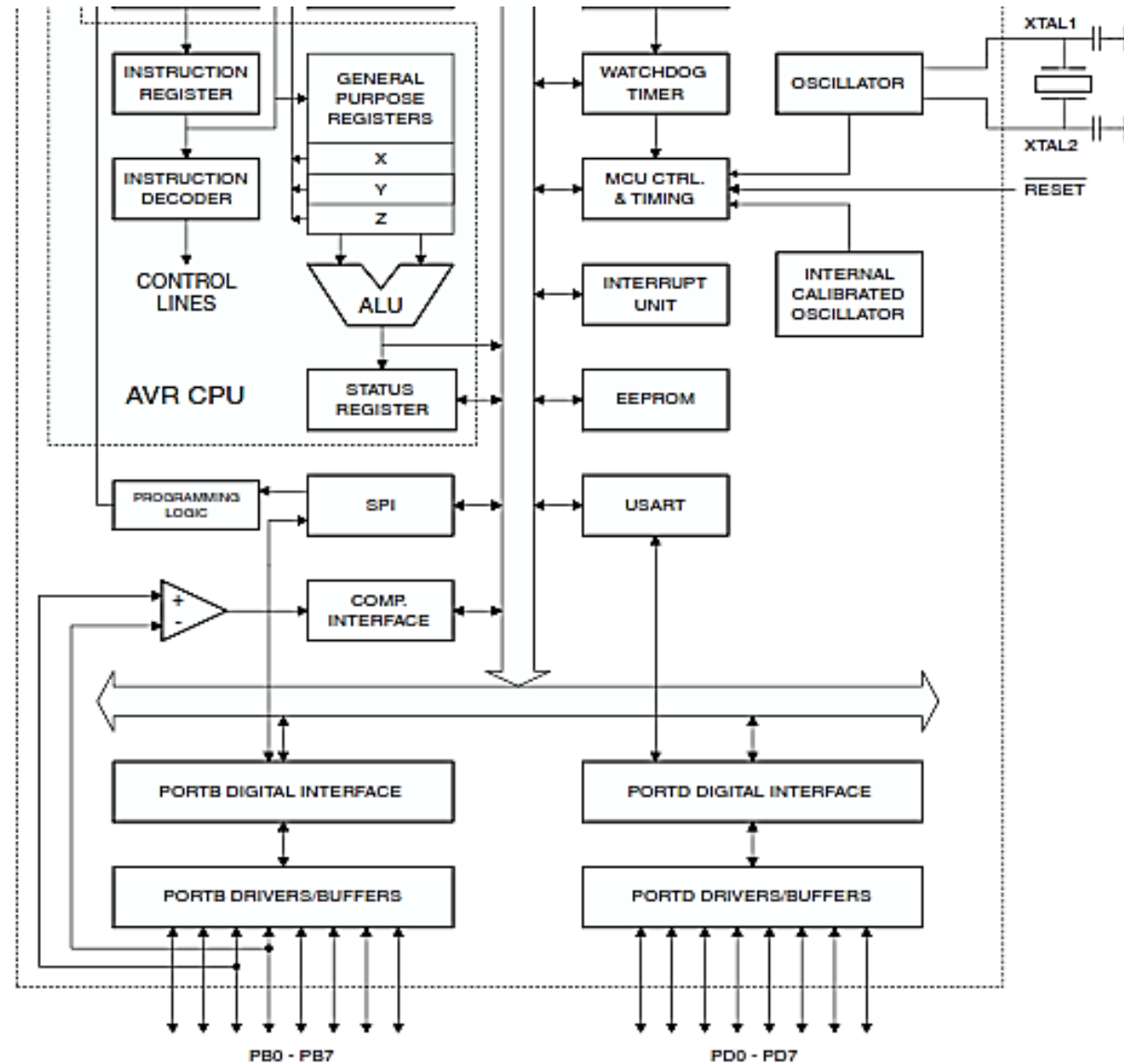
- ☐ **VCC (pin number 10)**
Digital supply voltage
- ☐ **GND (pin number 11 & 31)**
Ground
- ☐ **PORT A (PA7..PA0/ pin number 33..40)**
General I/O, Analog input for A/D Converter
- ☐ **PORT B (PB0..PB7/ pin number 22..29)**
General I/O, Timer/Counter, Ext. Interrupt, Analog Comparator, SPI
- ☐ **PORT C (PC0..PC7/ pin number 22..29)**
General I/O, TWI, JTAG, Timer oscillator
- ☐ **PORT D (PD0..PD7/ pin number 14..21)**
General I/O, USART, Ext. Interrupt, Timer/Counter
- ☐ **RESET (pin number 9)**
Reset MCU
- ☐ **XTAL1 & XTAL2 (pin number 12 & 13)**
Input and output of inverting oscillator
- ☐ **AVCC (pin number 30)**
Supply voltage pin for Port A and the A/D Converter
- ☐ **AREFF (pin number 32)**
Analog reference for A/D Converter



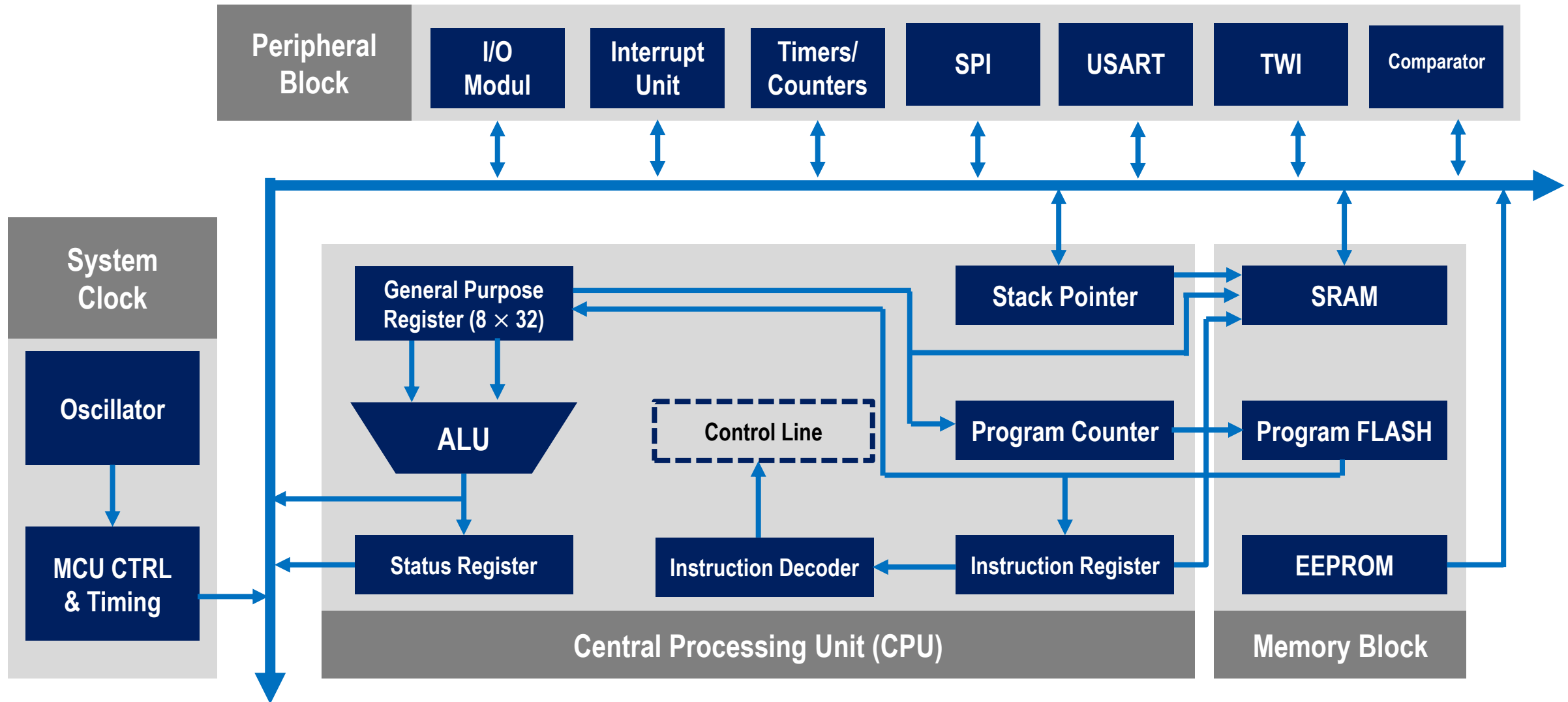
Blok Diagram ATmega16



Blok Diagram ATmega16 (Lanjutan)

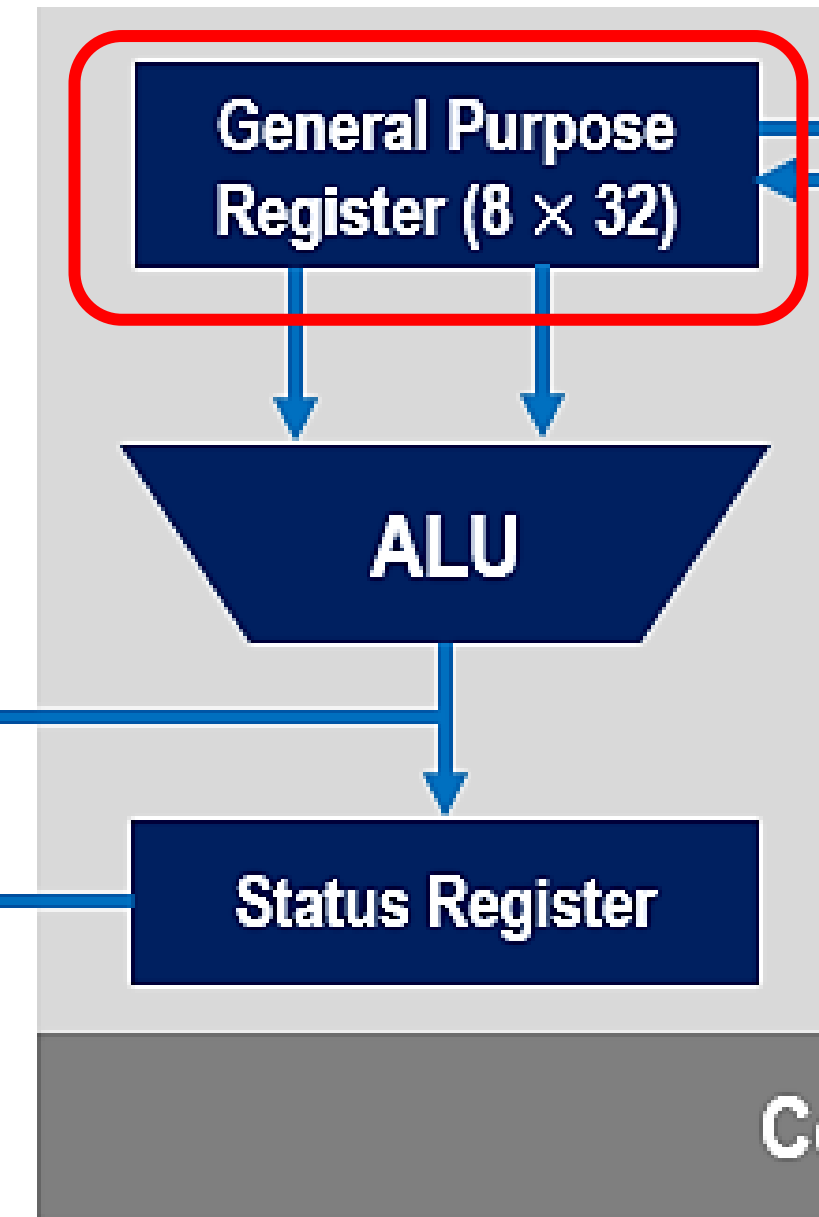


Arsitektur AVR (ATmega16)





Central Processing Unit



Register :

Sebuah slot yang berfungsi untuk menyimpan data, alamat, kode instruksi, dan bit status

General
Purpose
Working
Registers

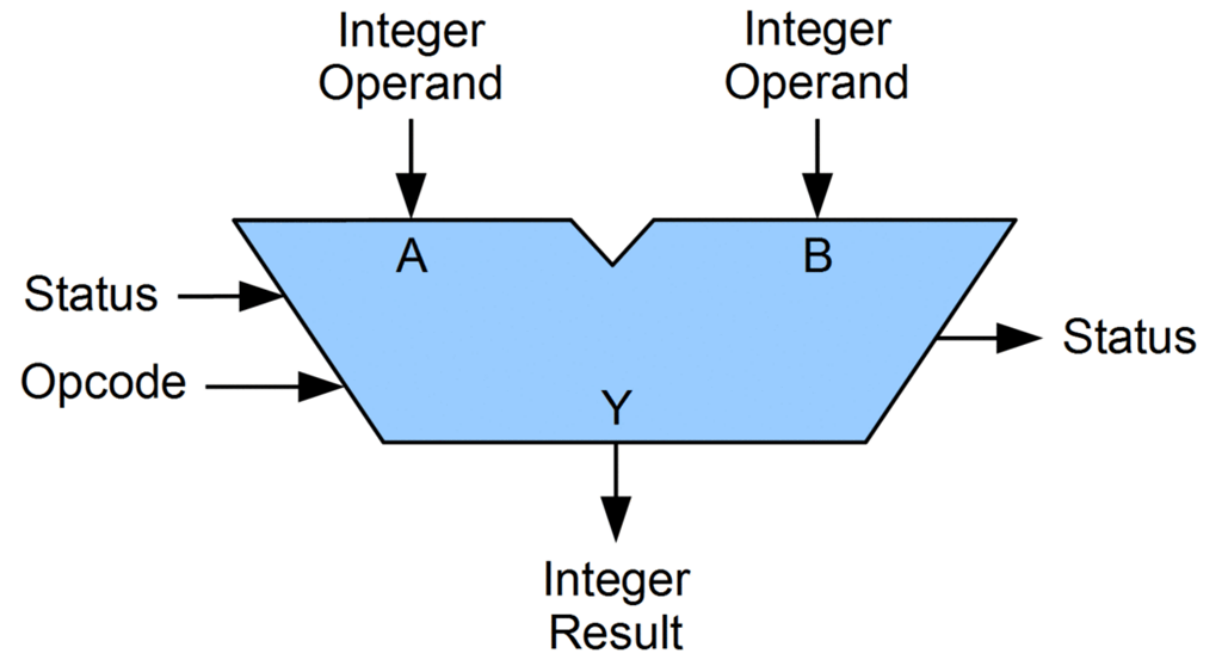
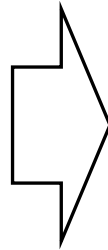
7	0	Addr.	
R0	0x00		
R1	0x01		
R2	0x02		
...			
R13	0x0D		
R14	0x0E		
R15	0x0F		
R16	0x10		
R17	0x11		
...			
R26	0x1A		X-register Low Byte
R27	0x1B		X-register High Byte
R28	0x1C		Y-register Low Byte
R29	0x1D		Y-register High Byte
R30	0x1E		Z-register Low Byte
R31	0x1F		Z-register High Byte

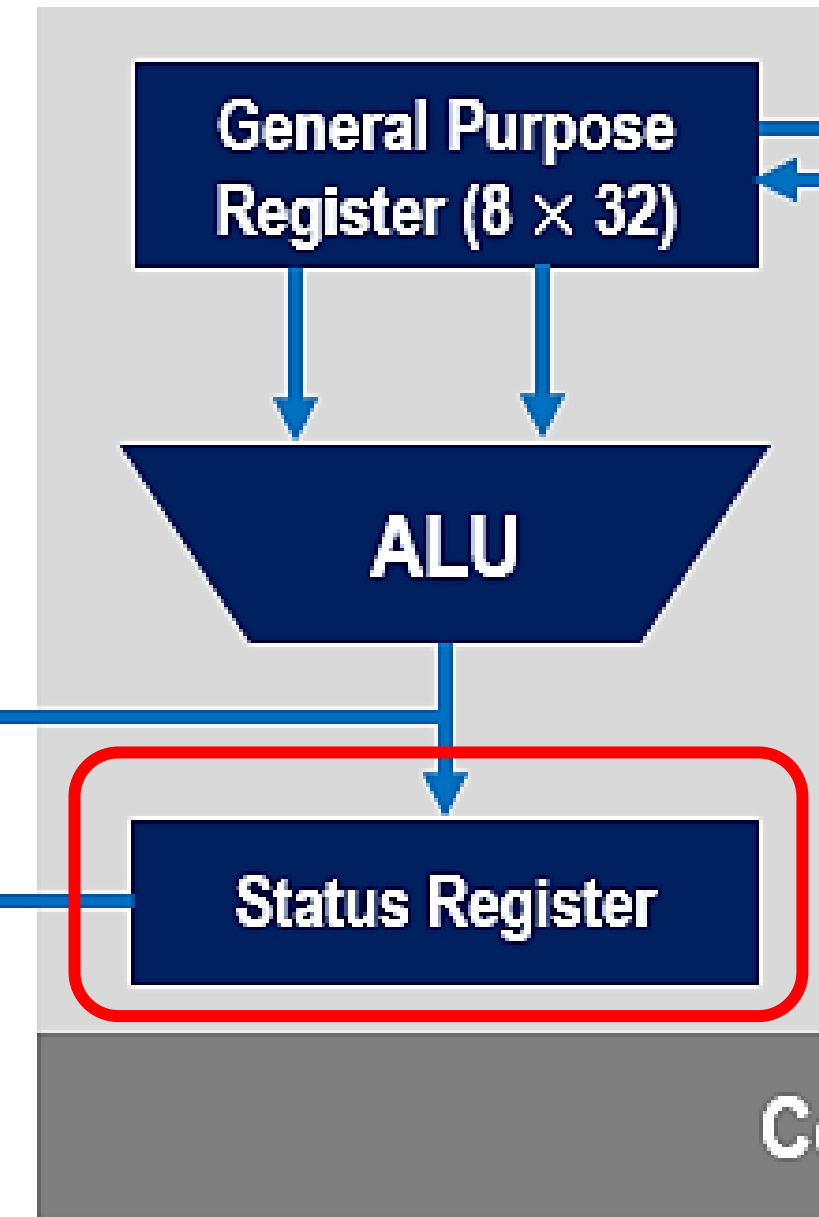
**General Purpose
Register (8 × 32)**

ALU

Status Register

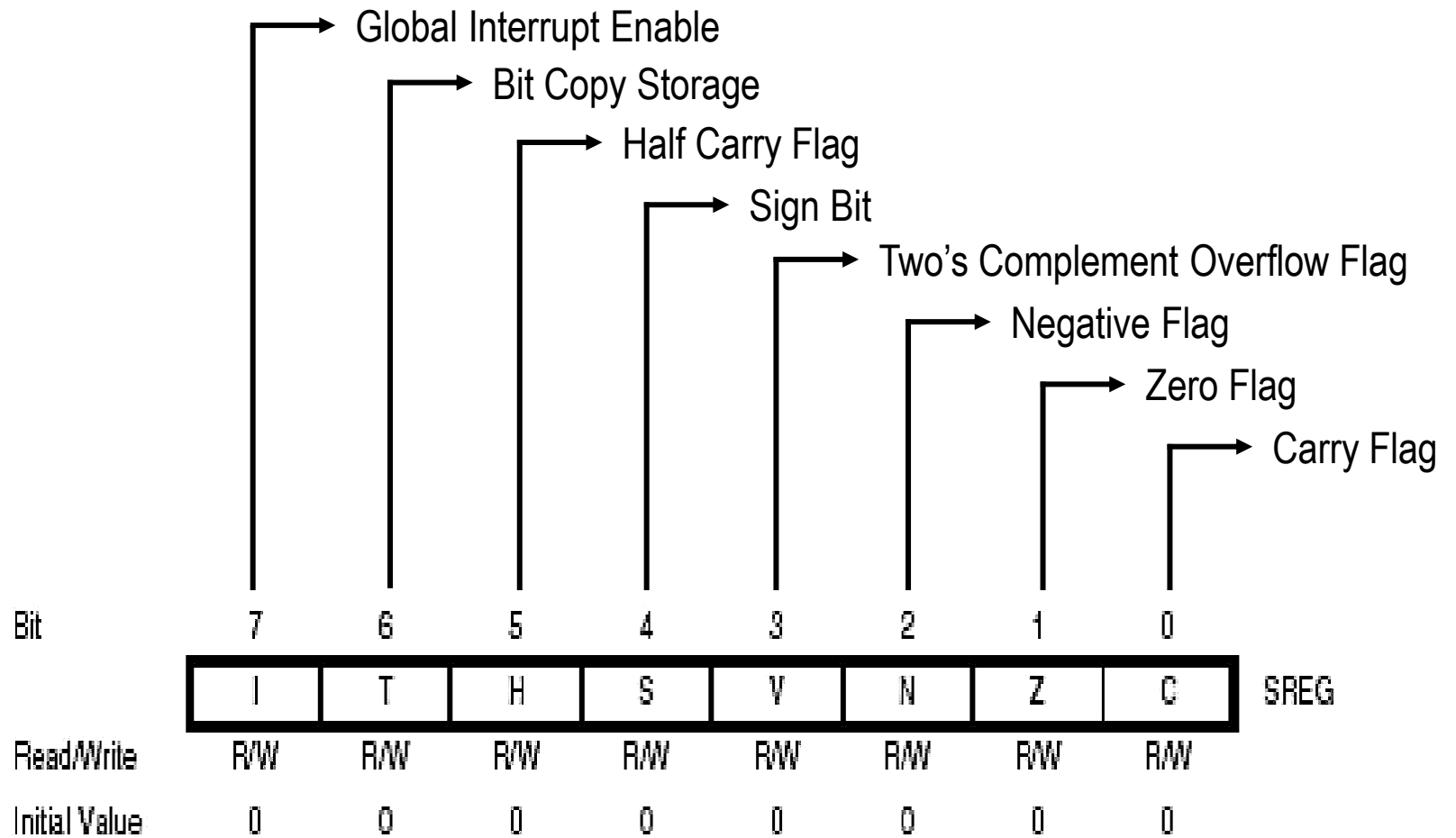
ALU (Arithmetic Logic Unit):
Berfungsi melakukan perhitungan
matematika dan logika.





Status Register:

Berisi informasi tentang hasil operasi operasi arithmetic logic unit.



Stack Pointer:

Umumnya digunakan untuk menyimpan data sementara, menyimpan local variables, dan menyimpan alamat kembali setelah interrupt.

Program Counter

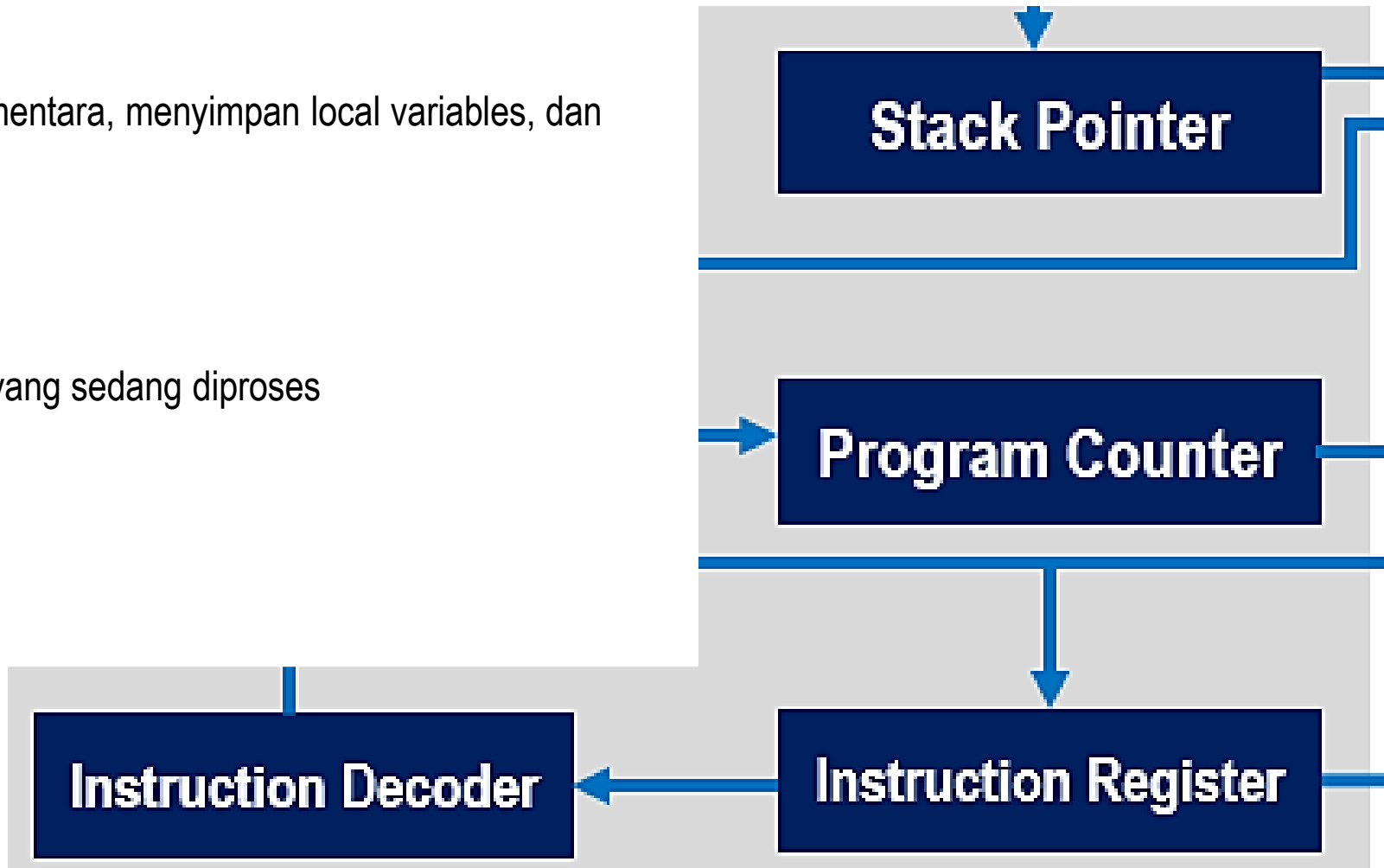
Menyimpan alamat memori yang berisi instruksi yang sedang diproses

Instruction Register

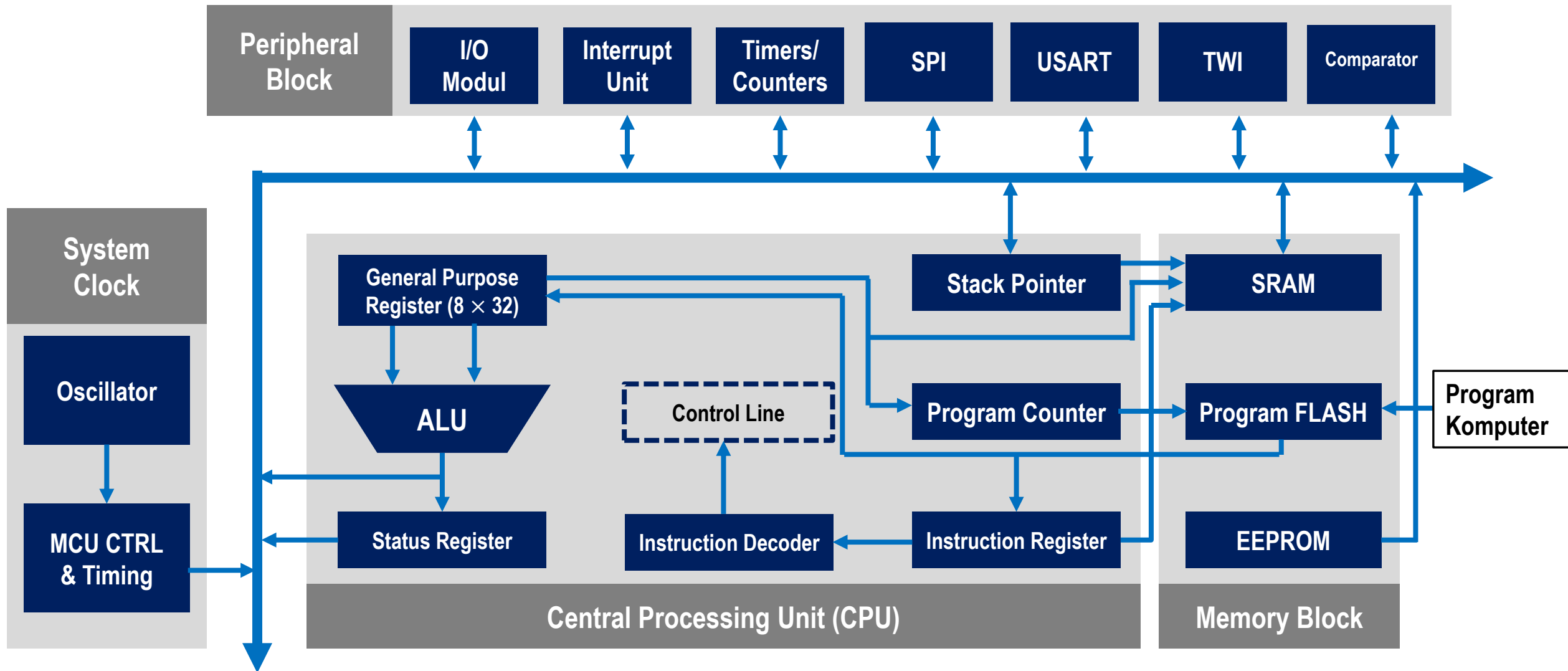
Berisi instruksi yang diproses oleh ALU

Instruction Decoder

Bertugas menerjemahkan instruksi dari instruction register



Arsitektur AVR (ATmega16)

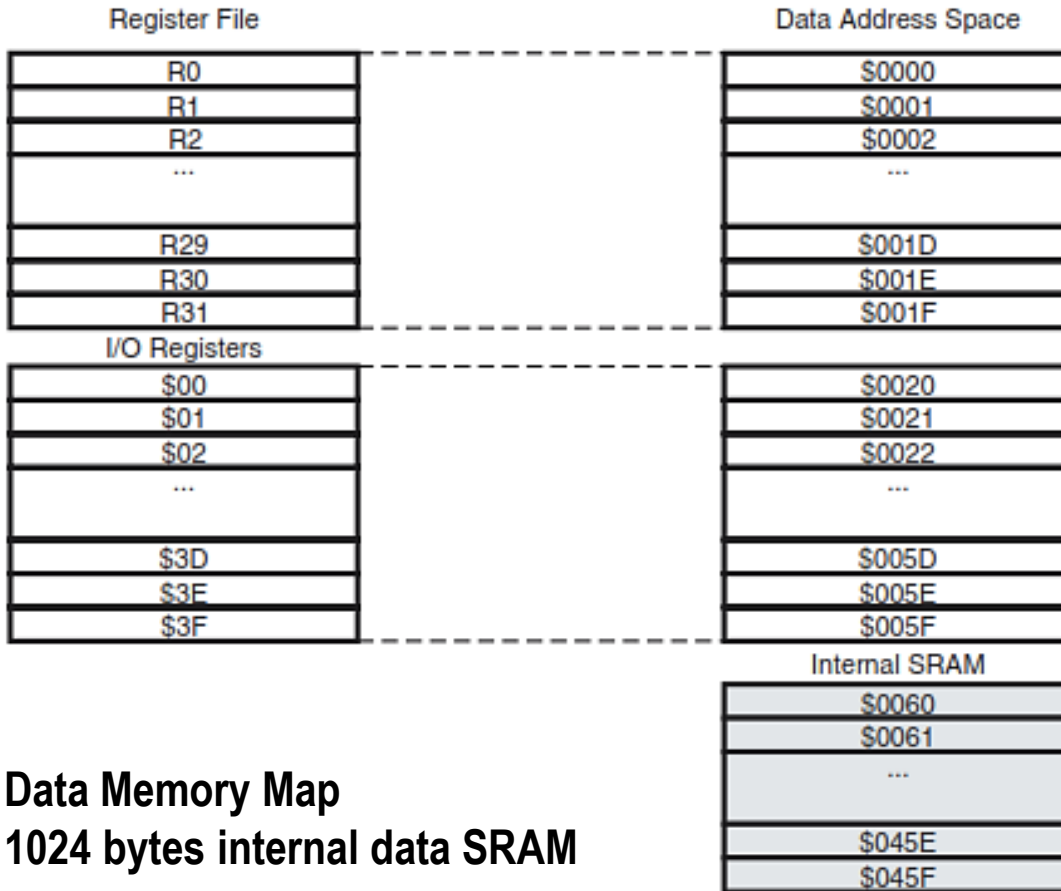




Memory Block

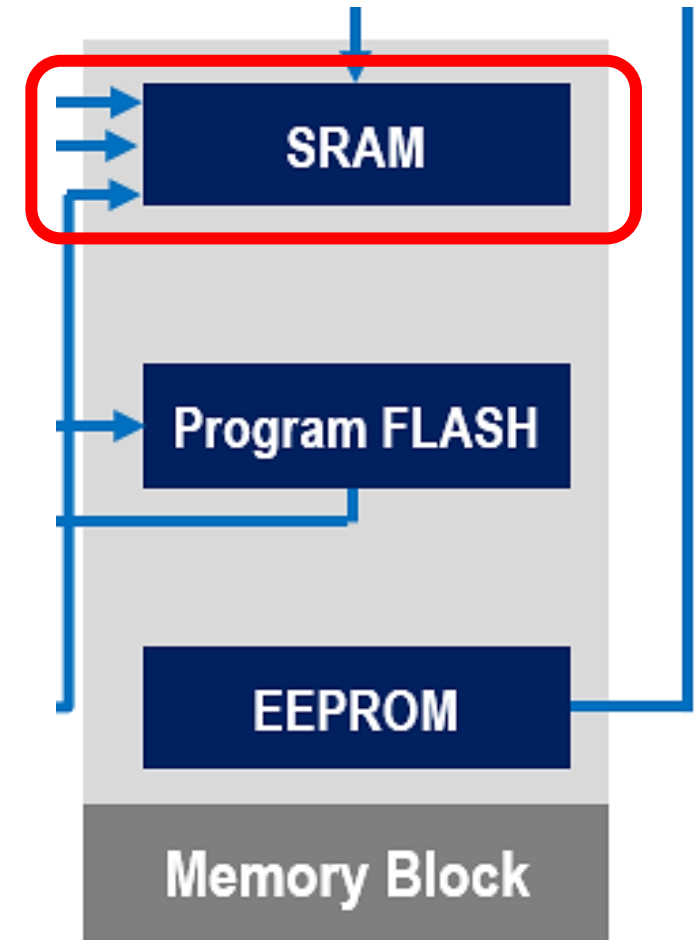
SRAM (Static Random Access Memory)

- Menyimpan program saat program berjalan
- Jenis volatile memory
- Penyimpanan sementara
- Tidak memerlukan refresh berkala seperti DRAM



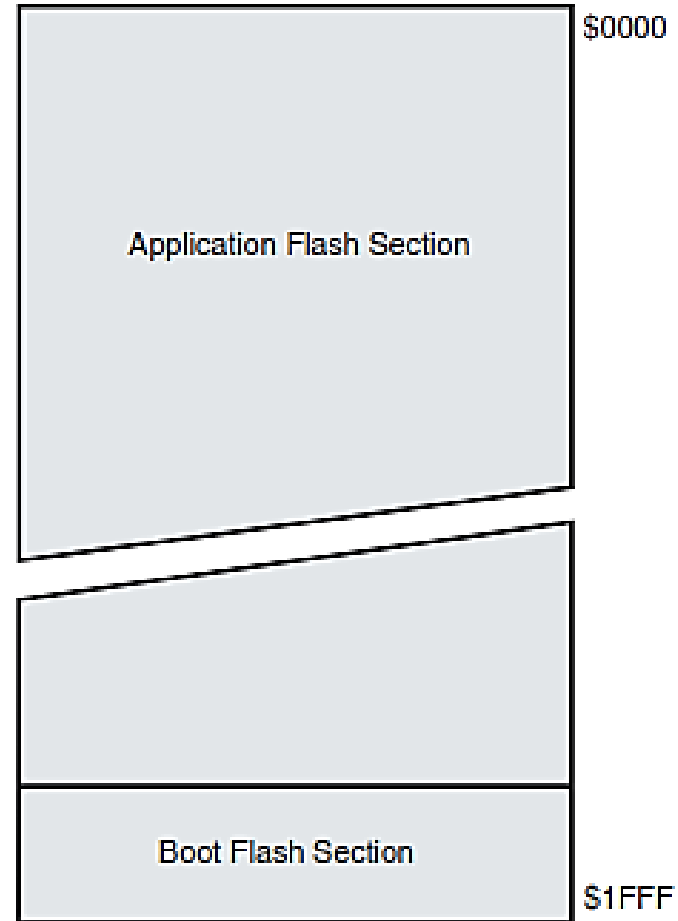
Data Memory Map

1024 bytes internal data SRAM

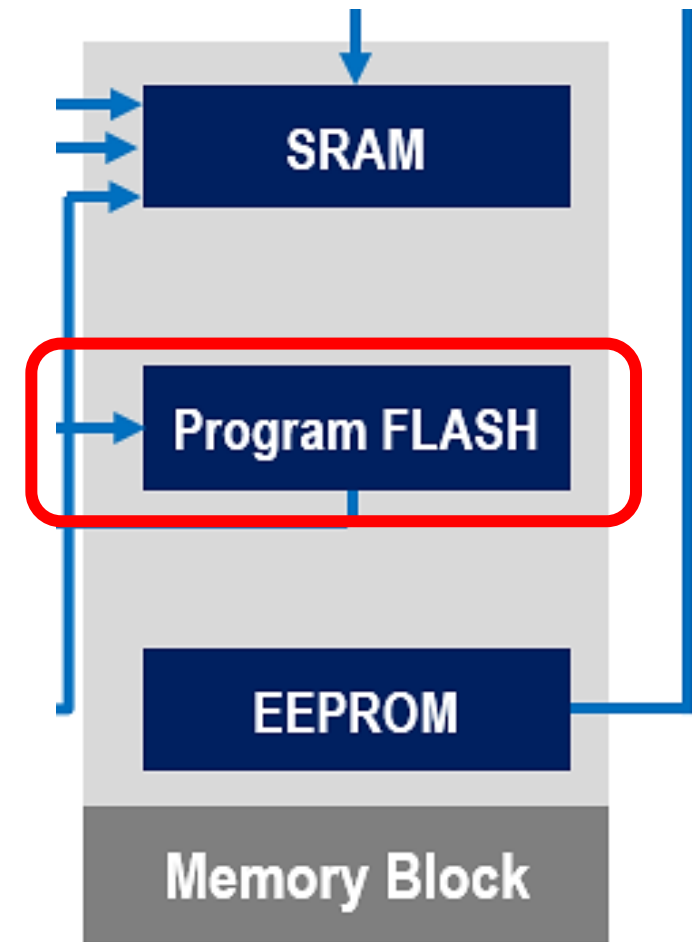


Program Flash

- Bertugas menyimpan kode program
- Bersifat non-volatile
- Program flash pada ATmega16 memiliki 10.000 kali siklus baca/tulis



Program Memory Map



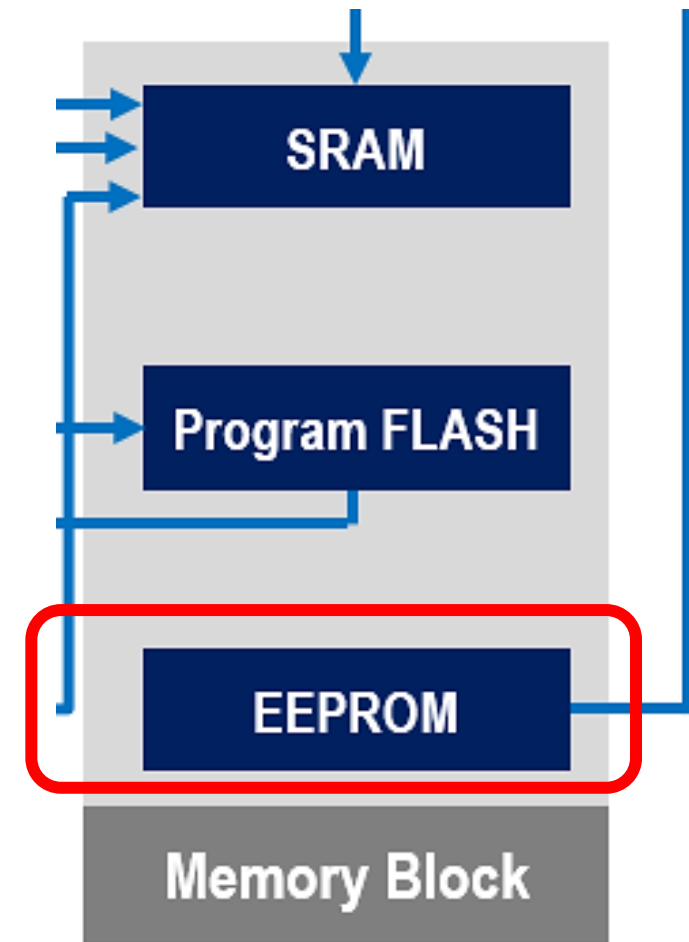
EEPROM (Electrically Erasable Programmable Read Only Memory)

- Bersifat non-volatile
- EEPROM pada ATmega16 memiliki 100.000 kali siklus baca/tulis

Bit	15	14	13	12	11	10	9	8	
	—	—	—	—	—	—	—	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	X	
	X	X	X	X	X	X	X	X	

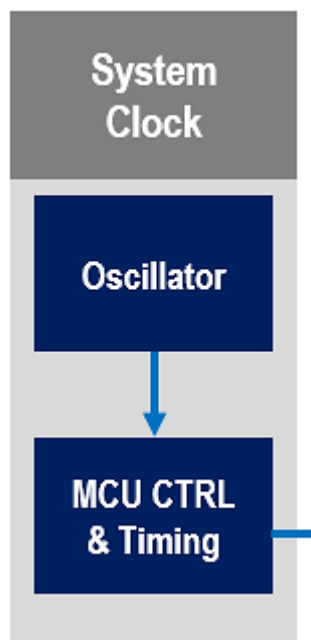
Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	EEDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	—	—	—	—	EERIE	EEMWE	EERE		EECR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	X	0	

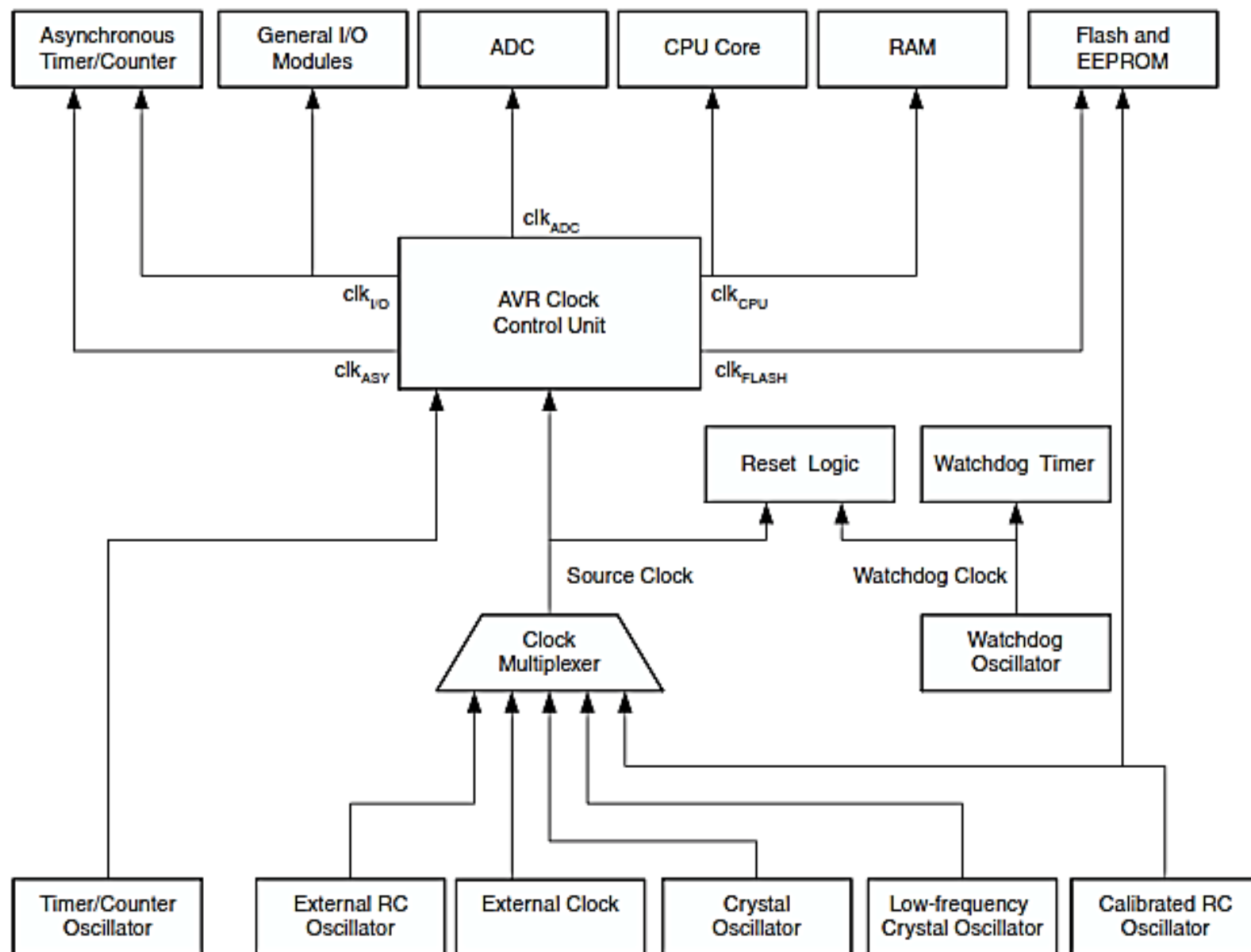




System Clock



Clock Distribution

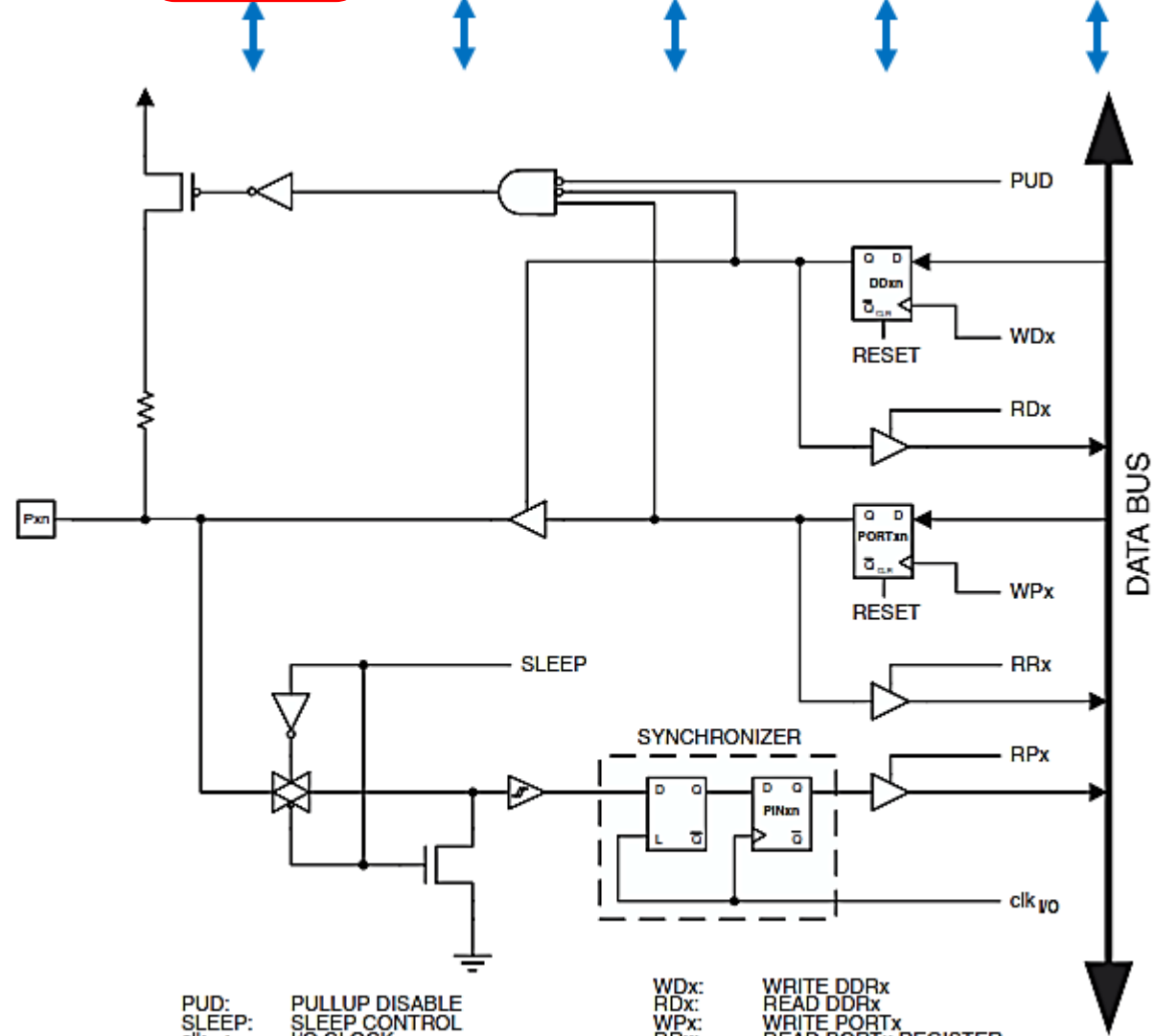


Device Clocking Option Select

Device Clocking Option	CKSEL3..0
External Crystal/Ceramic Resonator	1111 - 1010
External Low-frequency Crystal	1001
External RC Oscillator	1000 - 0101
Calibrated Internal RC Oscillator	0100 - 0001
External Clock	0000

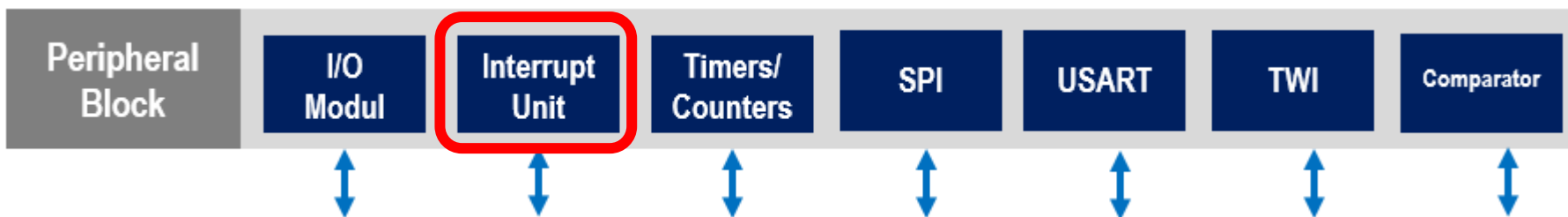


Peripheral Block



PUD: PULLUP DISABLE
SLEEP: SLEEP CONTROL
clk_{IO}: I/O CLOCK

WDx: WRITE DDRx
RDx: READ DDRx
WPx: WRITE PORTx
RRx: READ PORTx REGISTER
RPx: READ PORTx PIN



Reset and Interrupt Vectors

General Interrupt Control Register

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	INT2	—	—	—	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

■ Vector No

- An interrupt with a lower 'Vector No' will have a higher priority.
- E.g., INT0 has a higher priority than INT1 and INT2.

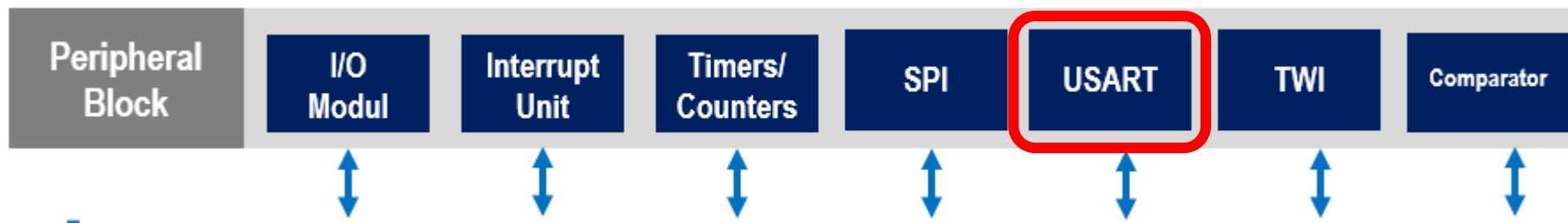
■ Program Address

- The fixed memory location for a given interrupt handler.
- E.g., in response to interrupt INT0, CPU runs instruction at \$002.

■ Interrupt Vector Name

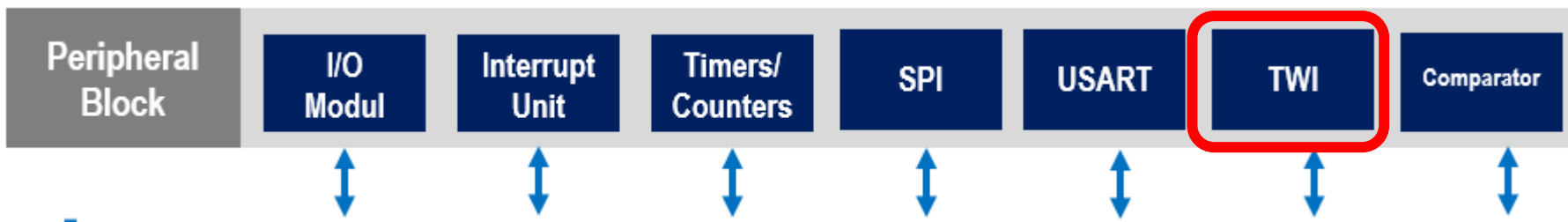
- This is the interrupt name, to be used with C macro ISR().

Vector No.	Program Address ⁽²⁾	Source	Interrupt Definition
1	\$000 ⁽¹⁾	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	\$002	INT0	External Interrupt Request 0
3	\$004	INT1	External Interrupt Request 1
4	\$006	TIMER2 COMP	Timer/Counter2 Compare Match
5	\$008	TIMER2 OVF	Timer/Counter2 Overflow
6	\$00A	TIMER1 CAPT	Timer/Counter1 Capture Event
7	\$00C	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	\$00E	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	\$010	TIMER1 OVF	Timer/Counter1 Overflow
10	\$012	TIMER0 OVF	Timer/Counter0 Overflow
11	\$014	SPI, STC	Serial Transfer Complete
12	\$016	USART, RXC	USART, Rx Complete
13	\$018	USART, UDRE	USART Data Register Empty
14	\$01A	USART, TXC	USART, Tx Complete
15	\$01C	ADC	ADC Conversion Complete
16	\$01E	EE_RDY	EEPROM Ready
17	\$020	ANA_COMP	Analog Comparator
18	\$022	TWI	Two-wire Serial Interface
19	\$024	INT2	External Interrupt Request 2
20	\$026	TIMER0 COMP	Timer/Counter0 Compare Match
21	\$028	SPM_RDY	Store Program Memory Ready



USART(Universal Synchronous Asynchronous Receiver/Transmitter)

- Mode sinkron dimana pengirim data mengeluarkan pulsa/clock untuk sinkronisasi data
- Mode asinkron, dimana pengirim data tidak mengeluarkan pulsa/clock, tetapi untuk proses sinkronisasi memerlukan inisialisasi, agar data yang diterima sama dengan data yang dikirimkan
- Pada proses inisialisasi, setiap perangkat yang terhubung harus memiliki baudrate yang sama

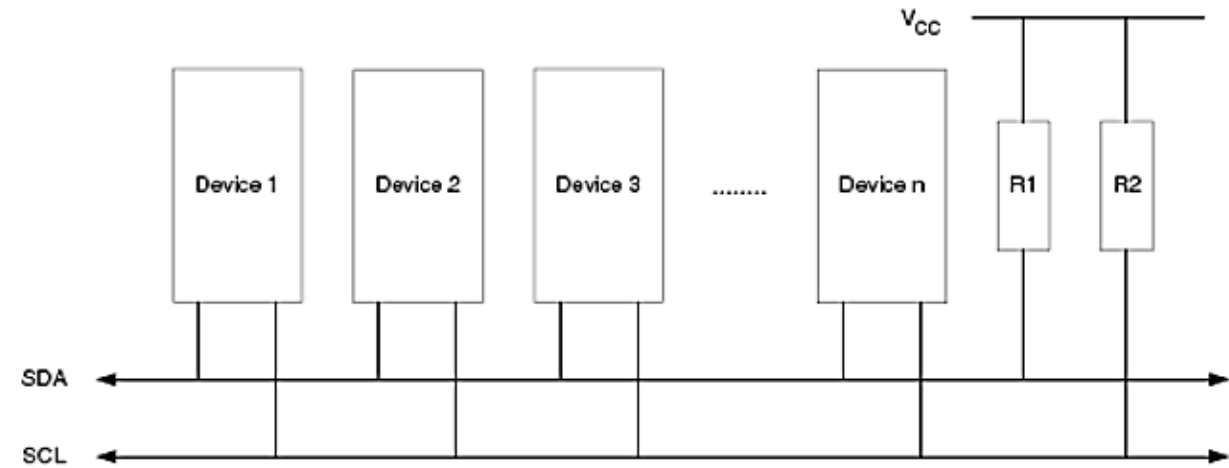


TWI (Two Wire Interface)

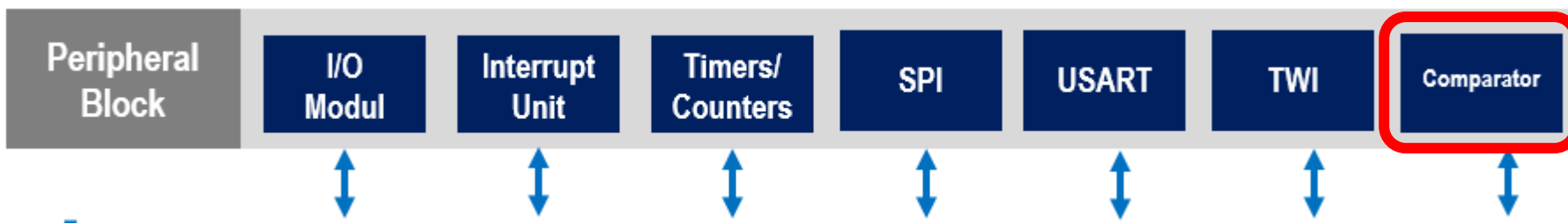
- Sebuah protokol untuk komunikasi serial antar IC
- Lebih fleksibel dari SPI

Term	Description
Master	The device that initiates and terminates a transmission. The master also generates the SCL clock.
Slave	The device addressed by a master.
Transmitter	The device placing data on the bus.
Receiver	The device reading data from the bus.

TWI Terminology

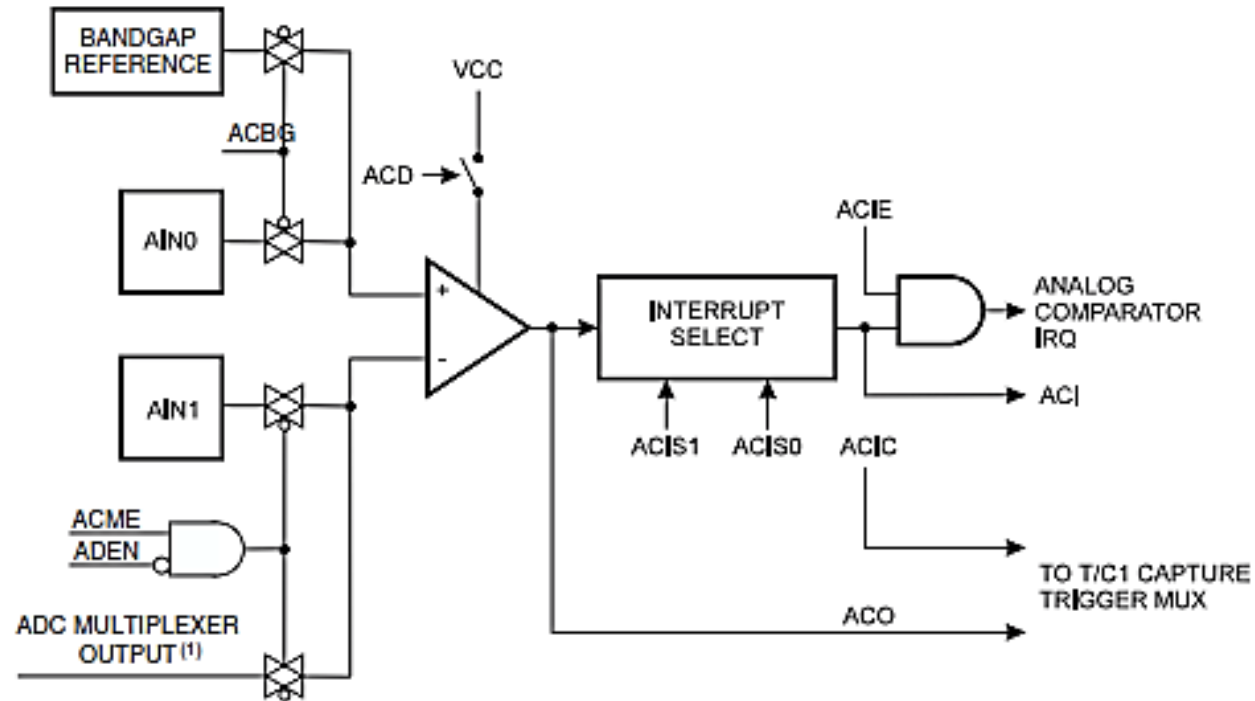


Typical TWI bus configuration



Comparator

- Untuk membandingkan tegangan input dengan sebuah referensi



Analog
Comparator
Block
Diagram

SEKIAN DAN TERIMA KASIH

PENGANTAR MIKROKONTROLER

Pertemuan 3

Ahmad Zarkasi

Tugas 1

Buatlah simulasi sederhana dengan menggunakan LED berbasis mikrokontroler ATmega dengan compiler CodeVision AVR.



CV
AVR



MATERI BAHASAN

PLATFORM ARDUINO

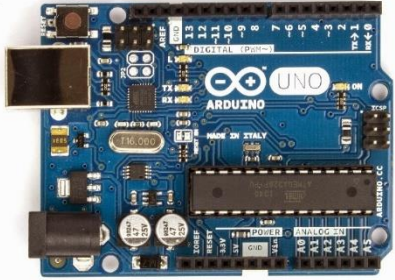
Apa itu Arduino?



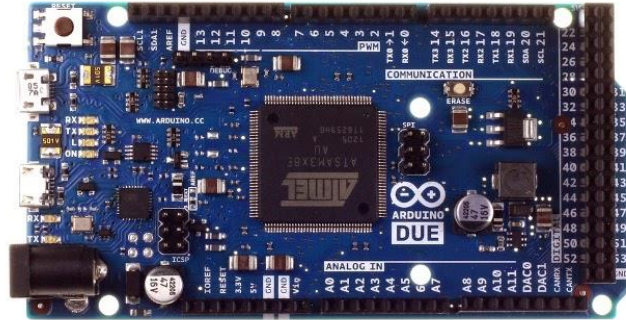
- Arduino is a microcontroller-based open source electronic prototyping board which can be programmed with an easy-to-use Arduino IDE.
- Arduino consists of both a physical programmable circuit board and a piece of software, or IDE. The Arduino IDE uses simplified version of C++, making it easier to learn.
- The Uno is one of the more popular boards in the Arduino family and great choice for beginners.

Beberapa Jenis Modul Arduino

Arduino Uno



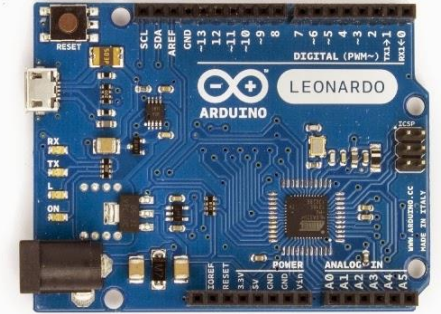
Arduino Due



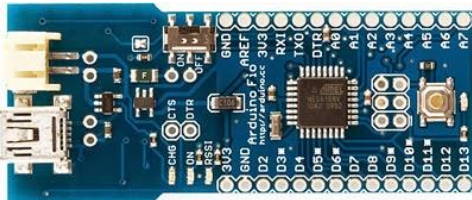
Arduino Mega



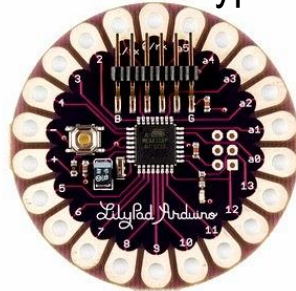
Arduino Leonardo



Arduino Fio



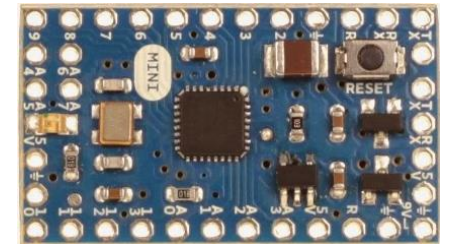
Arduino Lilypad



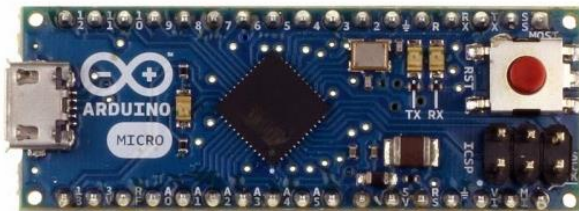
Arduino Nano



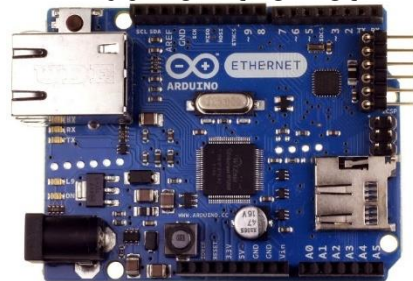
Arduino Mini



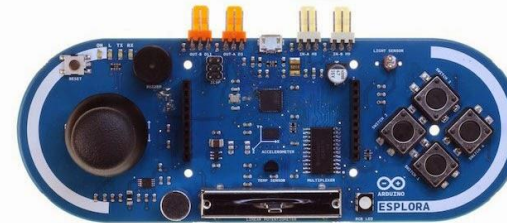
Arduino Micro



Arduino Ethernet



Arduino Esplora



Arduino Robot



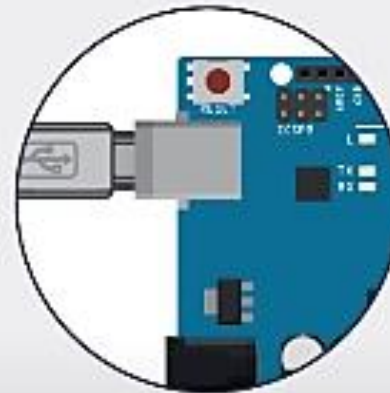
Bagian-bagian terpenting dari Arduino Uno



USB Connector



- ▶ This is a printer USB port used to load a program from the Arduino IDE onto the Arduino board.



Power Port



- ▶ The Arduino board can be powered through a AC-to-DC adapter or a battery



- ▶ The power source can be connected by plugging in a 2.1mm center-positive plug into the power jack of the board
- ▶ The Arduino UNO board operates at a voltage of 5 volts, but it can withstand a maximum voltage of 20 volts
- ▶ If the board is supplied with a higher voltage, there is a voltage regulator (it sits between the power port and USB connector) that protects the board from burning out

Microcontroller



- It is the most prominently visible black rectangular chip with 28 pins. Think of it as the brains of your Arduino



- The microcontroller used on the UNO board is Atmega328P by Atmel (a major microcontroller manufacturer)

Microcontroller

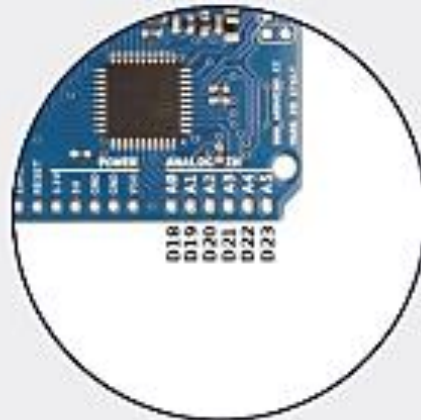


- ▶ Atmega328P has the following components in it
 - ▶ Flash memory of 32KB. The program loaded from the Arduino IDE is stored here
 - ▶ RAM of 2KB. This is runtime memory
 - ▶ CPU: It controls everything that goes on within the device. It fetches the program instructions from flash memory and runs it with the help of RAM
 - ▶ Electrically Erasable Programmable Read Only Memory (EEPROM) of 1KB. This is a type of nonvolatile memory, and it keeps the data even after device restart and reset
- ▶ Atmega328P is pre-programmed with bootloader. This allows you to directly upload a new Arduino program into the device, without using any external hardware programmer, making the Arduino UNO board easy to use

Analog input pins



- ▶ The Arduino UNO board has 6 analog input pins, labeled "Analog 0 to 5." These pins can read the signal from an analog sensor such as a temperature sensor and convert it into a digital value for system understanding



- ▶ These pins just measure voltage and not the current because they have very high internal resistance. Hence, only a small amount of current flows through these pins
- ▶ Although these pins are labeled analog and are analog input by default, these pins can also be used for digital input or output

Digital pins



- ▶ You can find these pins labeled “Digital 0 to 13.” These pins can be used as either input or output pins. When used as output, these pins act as a power supply source for the components connected to it and when used as input pins, they read the signals from the component connected to them



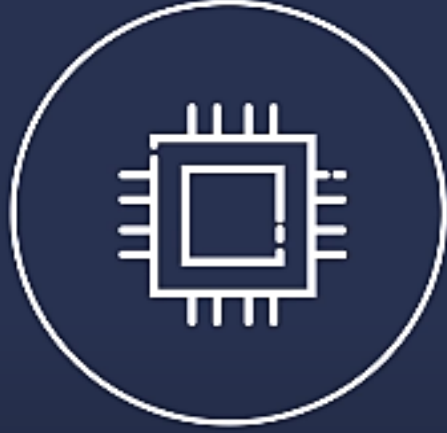
Crystal oscillator



- ▶ This is a quartz crystal oscillator which ticks 16 million times a second. On each tick, the microcontroller performs one operation, for example, addition, subtraction, etc



USB interface chip



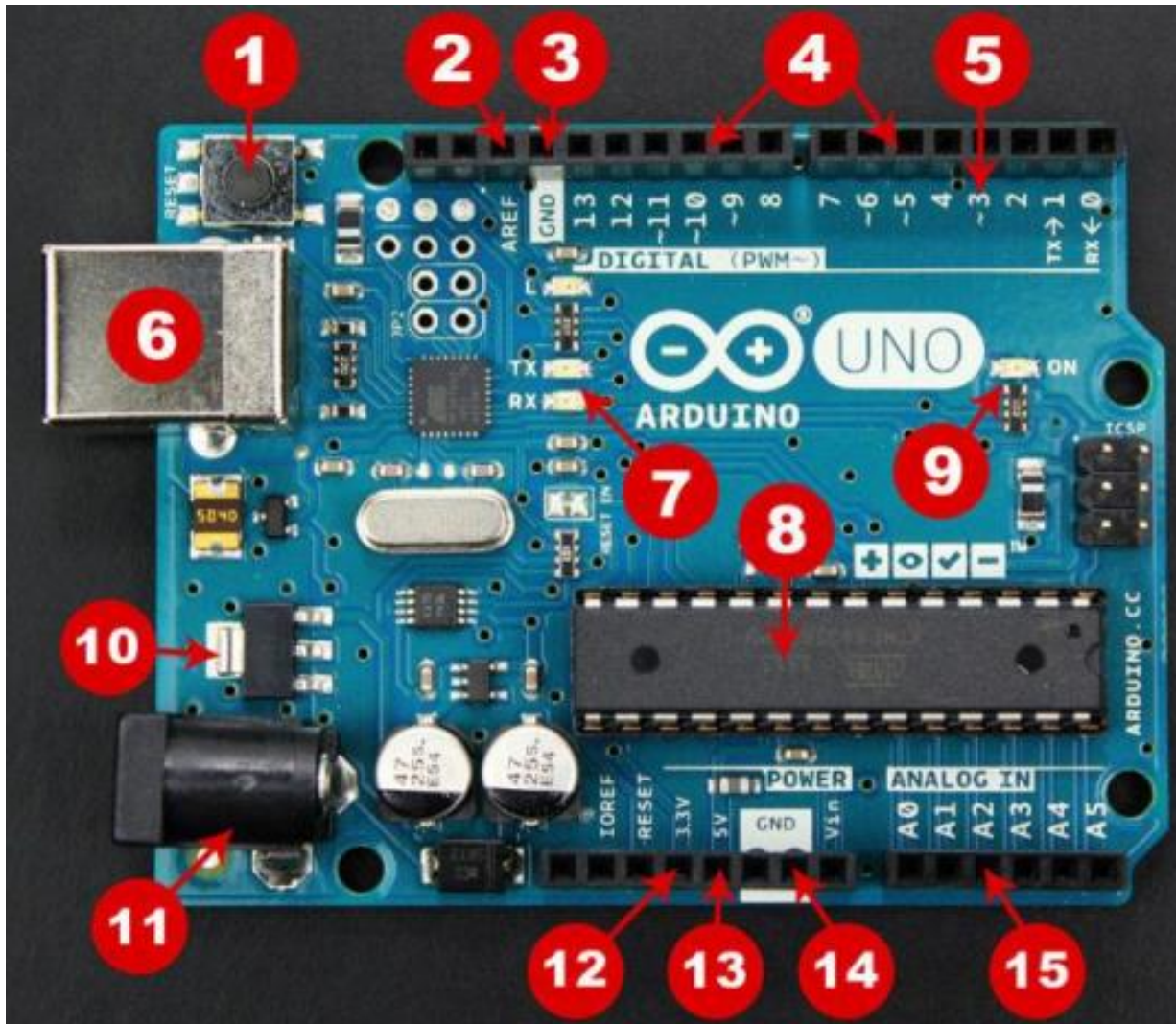
- Think of this as a signal translator. It converts signals in the USB level to a level that an Arduino UNO board understands

TX RX indicator



- TX stands for transmit, and RX for receive. These are indicator LEDs which blink whenever the UNO board is transmitting or receiving data

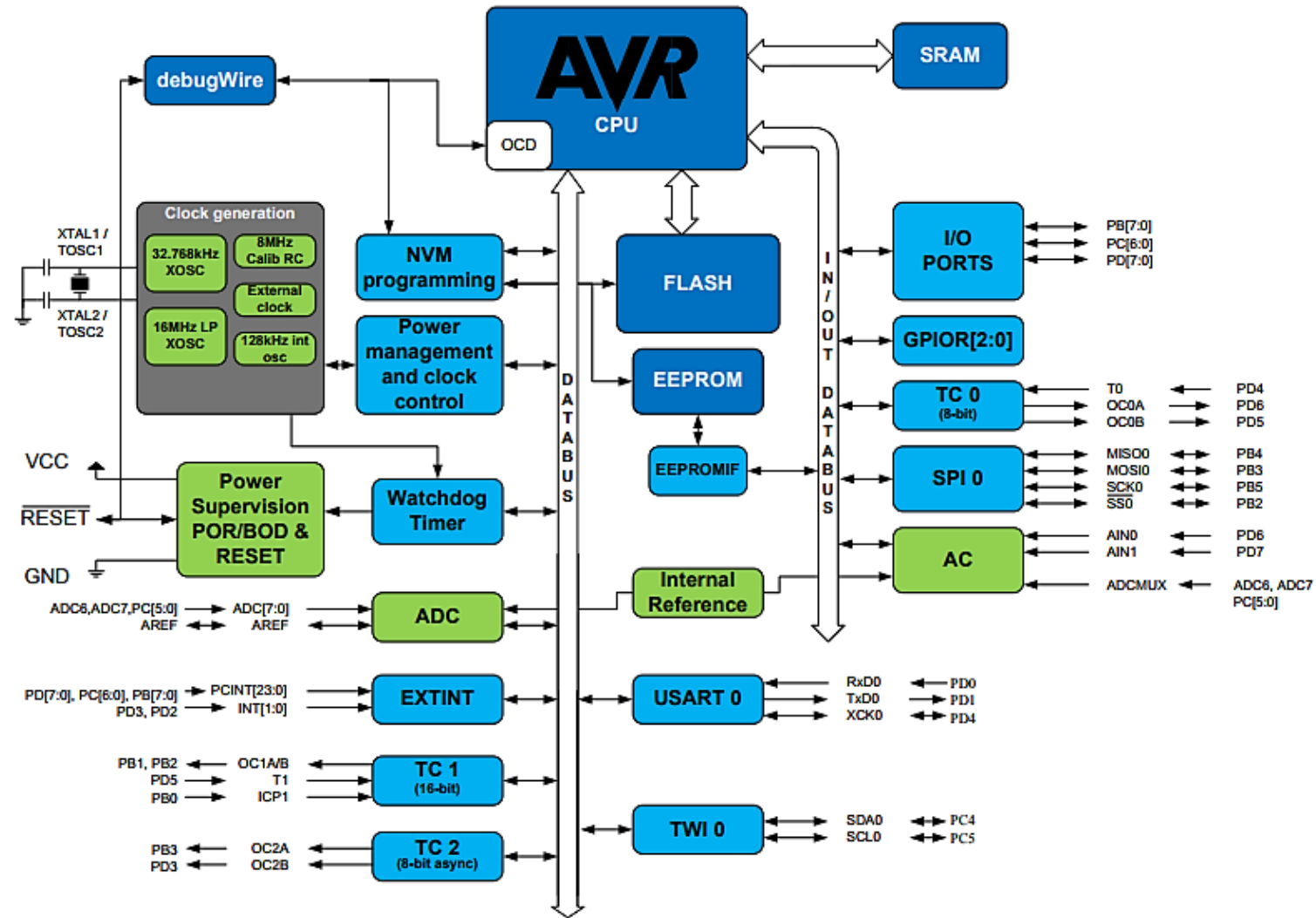
Ringkasan Bagian-bagian Arduino Uno



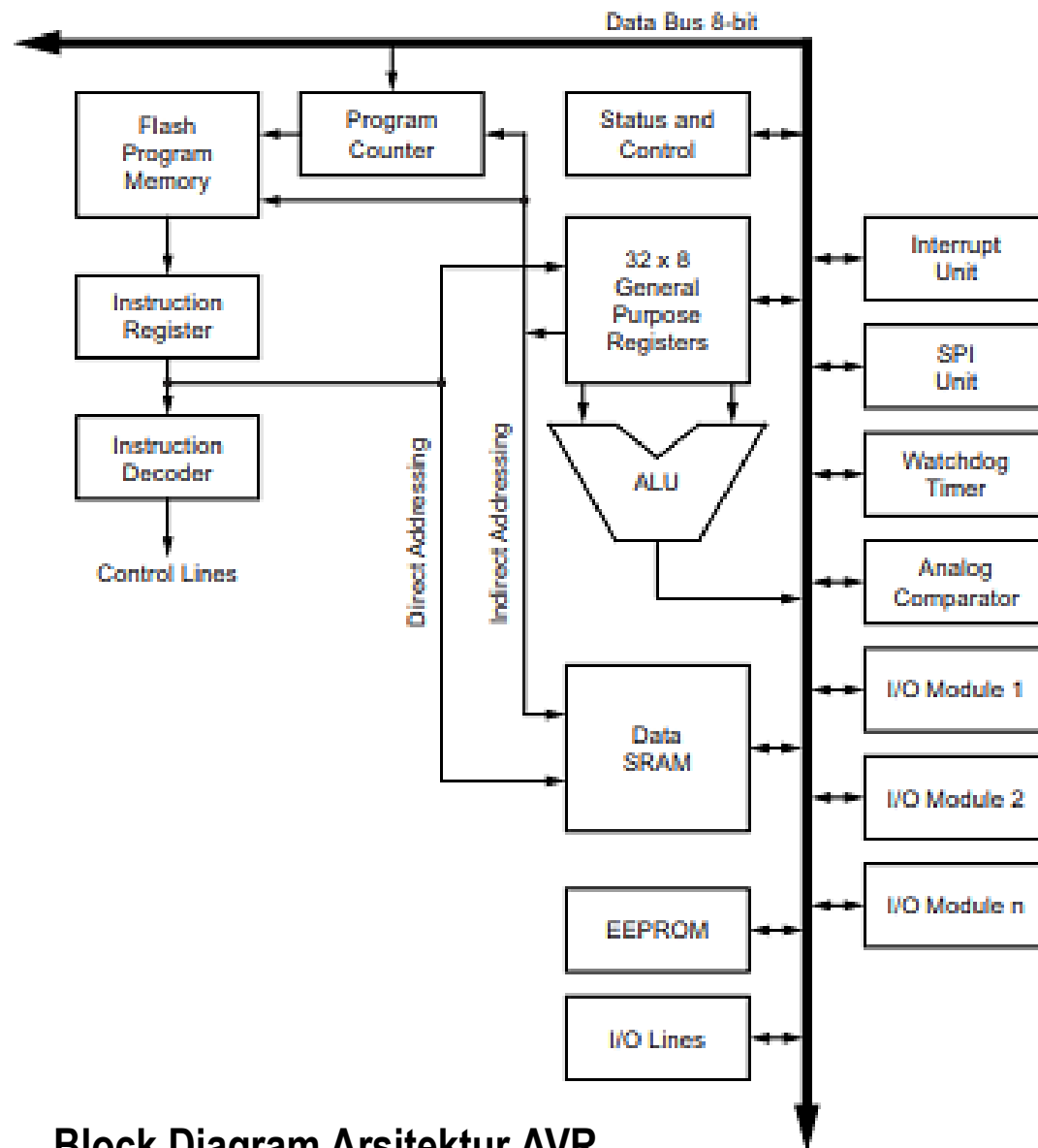
1. Reset Button
2. AREF
3. Ground Pin
4. Digital I/O
5. PWM
6. USB Connection
7. TX/RX
8. ATmega328P
9. Power LED Indicator
10. Voltage Regulator
11. DC Power Barrel Jack
12. 3.3V Pin
13. 5V Pin
14. Ground Pin
15. Analog Pin

ATmega328P

Block
Diagram



ATmega328P



Block Diagram Arsitektur AVR

Tugas Simulasi

Buatlah sebuah project simulasi menggunakan Proteus sebagai media desain dan Arduino IDE sebagai compiler.

Penilaian:

- Tingkat kesulitan dan kreativitas.
- Presentasi



SEKIAN DAN TERIMA KASIH

PENGANTAR MIKROKONTROLER

Pertemuan 4

Ahmad Zarkasi

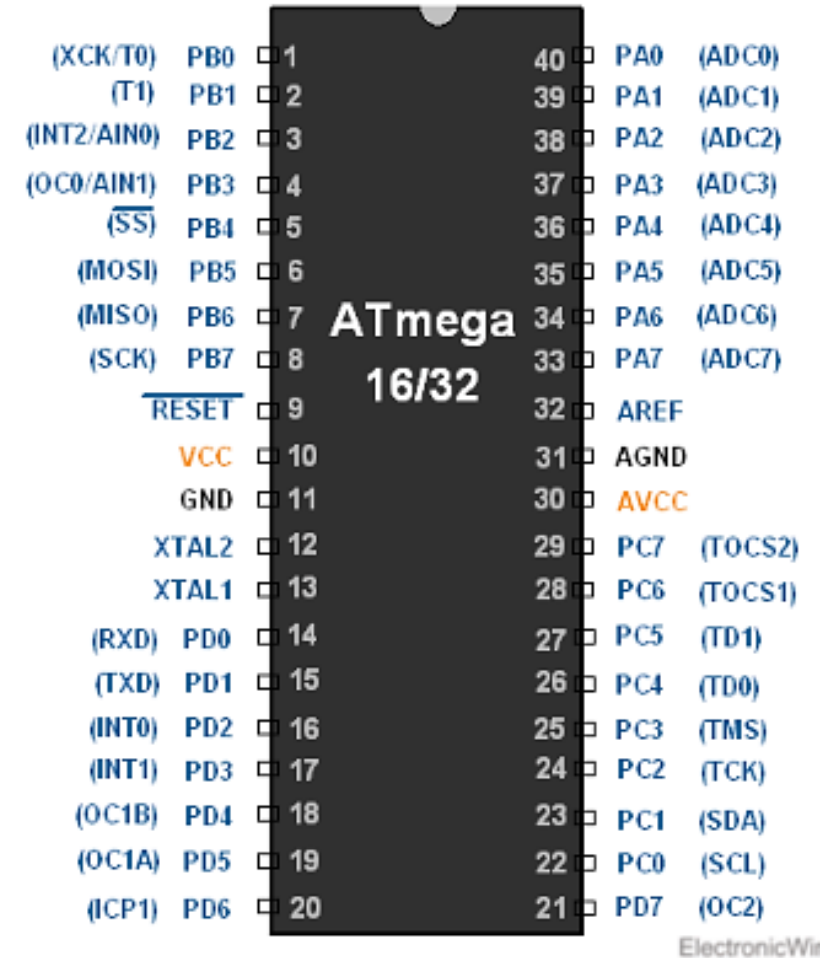


MATERI BAHASAN

PORT GPIO & REGISTER PADA AVR

Review

- AVR ATmega16 has 32 pins constituting four ports. The ports are listed below:
 1. PORT A
 2. PORT B
 3. PORT C
 4. PORT D
- Each pins of these four ports can be used as general-purpose inputs/outputs
- These pins can be configured as input or output using the three I/O registers for each port. These registers are listed below:
 1. DDRxregisters
 2. PINxregisters
 3. PORTxregisters(where x can be A, B, C, or D depending on which port registers are being addressed).
- Each pin also has some special functionality associated with it.



Pin Diagram of ATmega 16/32

DDRx (Data Direction Registers)

- These are 8-bit registers.
- These are used to configure the pins of the ports as input or output.
- Writing a one to the bits in this register sets those specific pins as output pins.
- Writing a zero to the bits in this register sets those specific pins as input pins.
- All bits in these registers can be read as well as written to.
- The initial value of these bits is zero.

Example:

1. Setting Port D as an output port:
`DDRD = 0xFF; atau DDRD=0b11111111;`
2. Setting Port D as input port:
`DDRD = 0x00; DDRD=0b00000000;`

PORTx : Data Registers

- These are 8-bit registers.
- These are used to put pins of the ports in a logic HIGH or logic LOW state.
- Writing a one to the bits in this register puts a LOW logic (0V) on those pins.
- All bits in these registers can be read as well as written to.
- The initial value of these bits is zero

Example:

We will use the PORTx register of Port D to write a value 0x55 to Port D

```
PORTD = 0x55;
```

PINx : Input Pins Address Registers

- These are 8-bit registers.
- These are used to read the values on the specific pins of the port.
- These bits are read-only bits and cannot be written to.

Example:

We will read the value on Port D in an 8-bit variable named 'port_value'

```
Port_value = PIND;
```

In the figure shown below, the above mentioned three registers for Port A are shown. These 8-bit registers.

7	6	5	4	3	2	1	0
PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0

PORTA

7	6	5	4	3	2	1	0
DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0

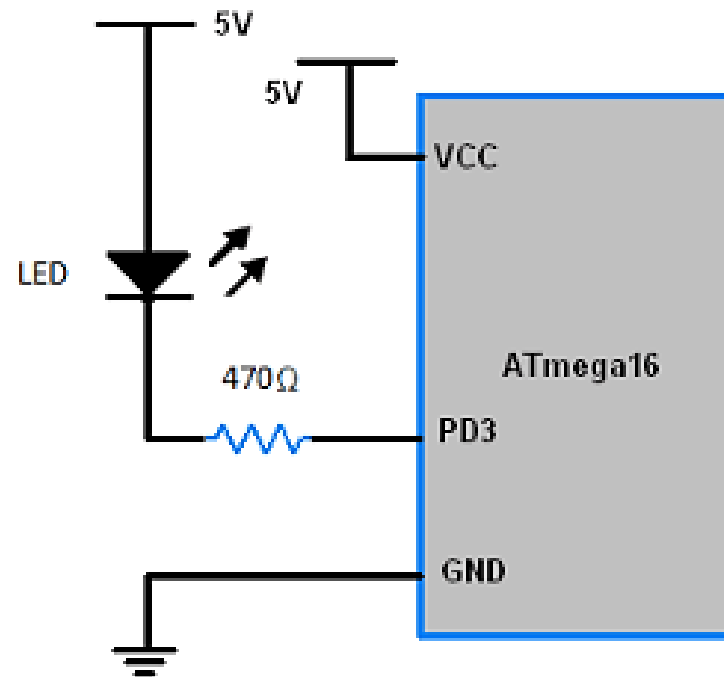
DDRA

7	6	5	4	3	2	1	0
PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0

PINA

Exercise

Lets write a code to program pin 3 of Port D as an output and use it to drive an LED. We sill toggle the LED with some delay.





MATERI BAHASAN

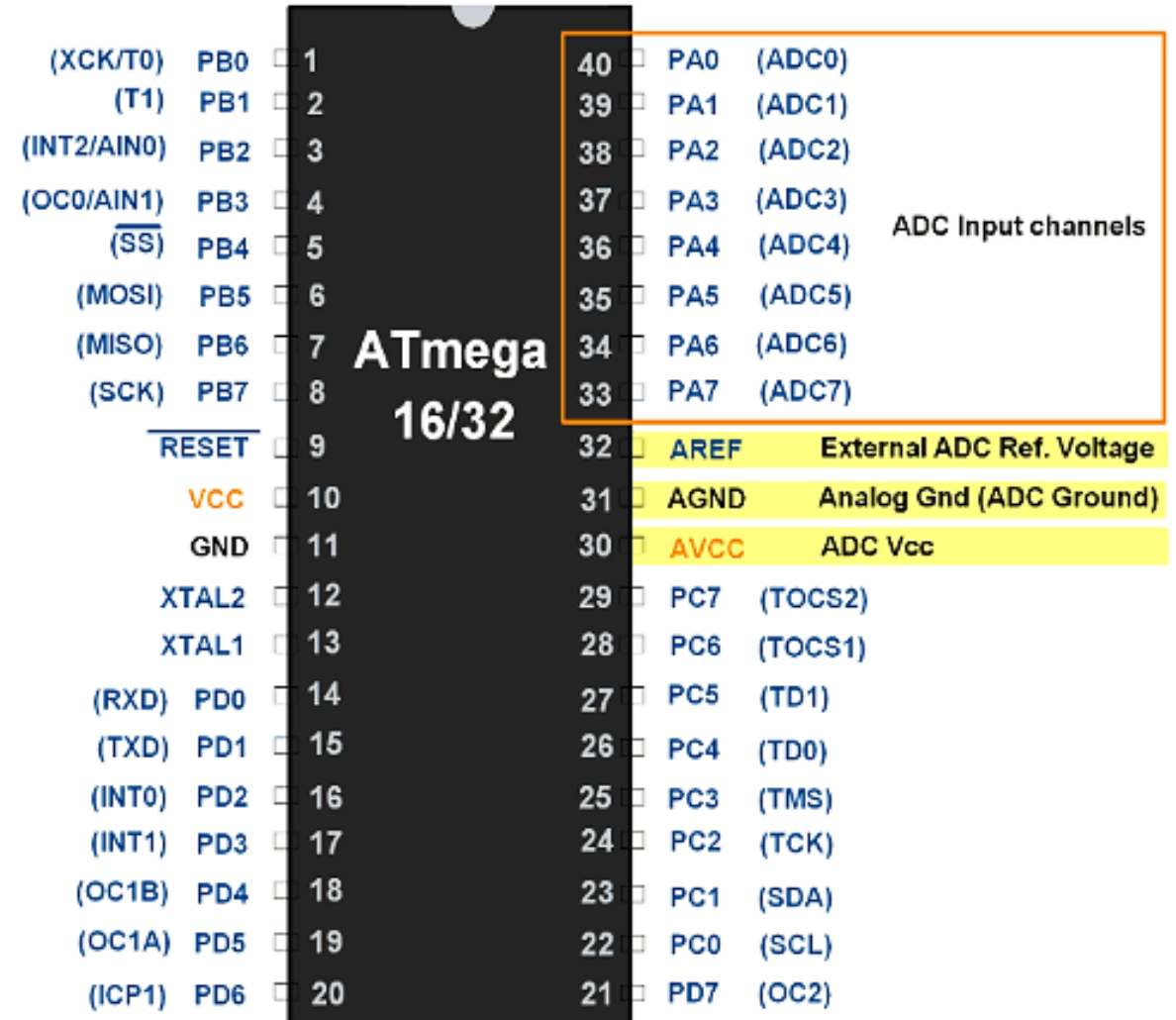
ADC pada AVR ATmega16

Definition

ADC (Analog to Digital Converter) is the most widely used device in embedded systems which is designed especially for data acquisition. In the AVR ATmega series normally 10-bit ADC is inbuilt in the controller.

Let us see how to use the ADC of AVR ATmega16 / ATmega32

Atmega16/32 supports eight ADC channels, which means we can connect eight analog input at a time. ADC channel 0 to channel 7 are present on PORT A. i.e. Pin no. 33 to 40.



ADC Pins of ATmega 16/32

Resolution

The controller has 10 bit ADC, which means we will get digital output 0 to 1023. i.e. when the input is 0V, the digital output will be 0V & when input is 5V (and $V_{ref}=5V$), we will get the highest digital output corresponding to 1023 steps, which is 5V.

So controller ADC has 1023 steps and

- Step size with $V_{ref} = 5V$, so $5/1023 = 4,88 \text{ mV}$
- Step size with $V_{ref}=2.56V$, so $2,56/1023 = 2,5 \text{ mV}$

So Digital data output will be $D_{out} = V_{in} / \text{step_size}$.

Specifications (ATmega16/32 ADC)

- It is 10-bit ADC.
- Converted output binary data is held in two special function 8-bit register ADCL (result Low) and ADCH (result in HIGH).
- ADC gives 10-bit output, so (ADCH:ADCL) only 10-bits are useful out of 16-bits.
- We have options to use this 10-bits as upper bits or lower bits.
- We also have three options for Vref
 1. AVcc (analog Vcc)
 2. Internal 2,56 V
 3. External Aref Pin
- If you decide to use Avcc or Vref pins as ADC voltage reference, you can make it more stable and increase the precision of ADC by connecting a capacitor between that pin and GND.

ADC Register

In AVR ADC, we need to understand four main register

1. **ADCH**: Holds digital converted data higher byte
2. **ADCL**: Holds digital converted data lower byte
3. **ADMUX**: ADC Multiplexer selection register
4. **ADCSRA**: ADC Control and status register

ADMUX Register

7	6	5	4	3	2	1	0
REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

ADCSRA Register:

7	6	5	4	3	2	1	0
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

SEKIAN DAN TERIMA KASIH

PENGANTAR MIKROKONTROLER

Pertemuan 5

Ahmad Zarkasi

MATERI BAHASAN

TIMER/COUNTER

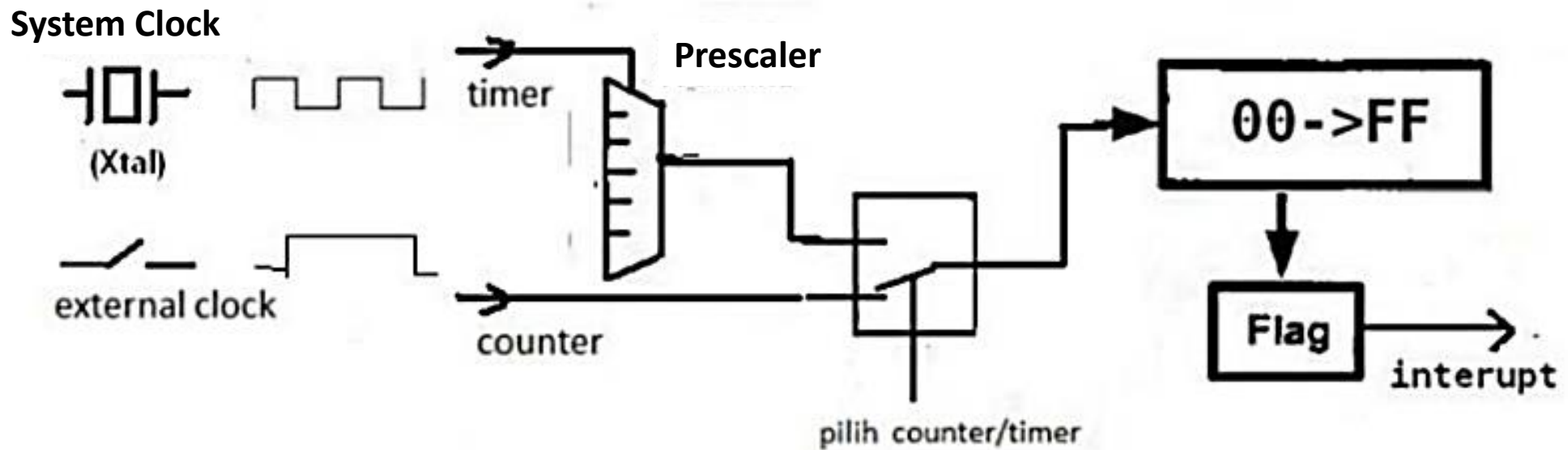
CAPAIAN PEMBELAJARAN

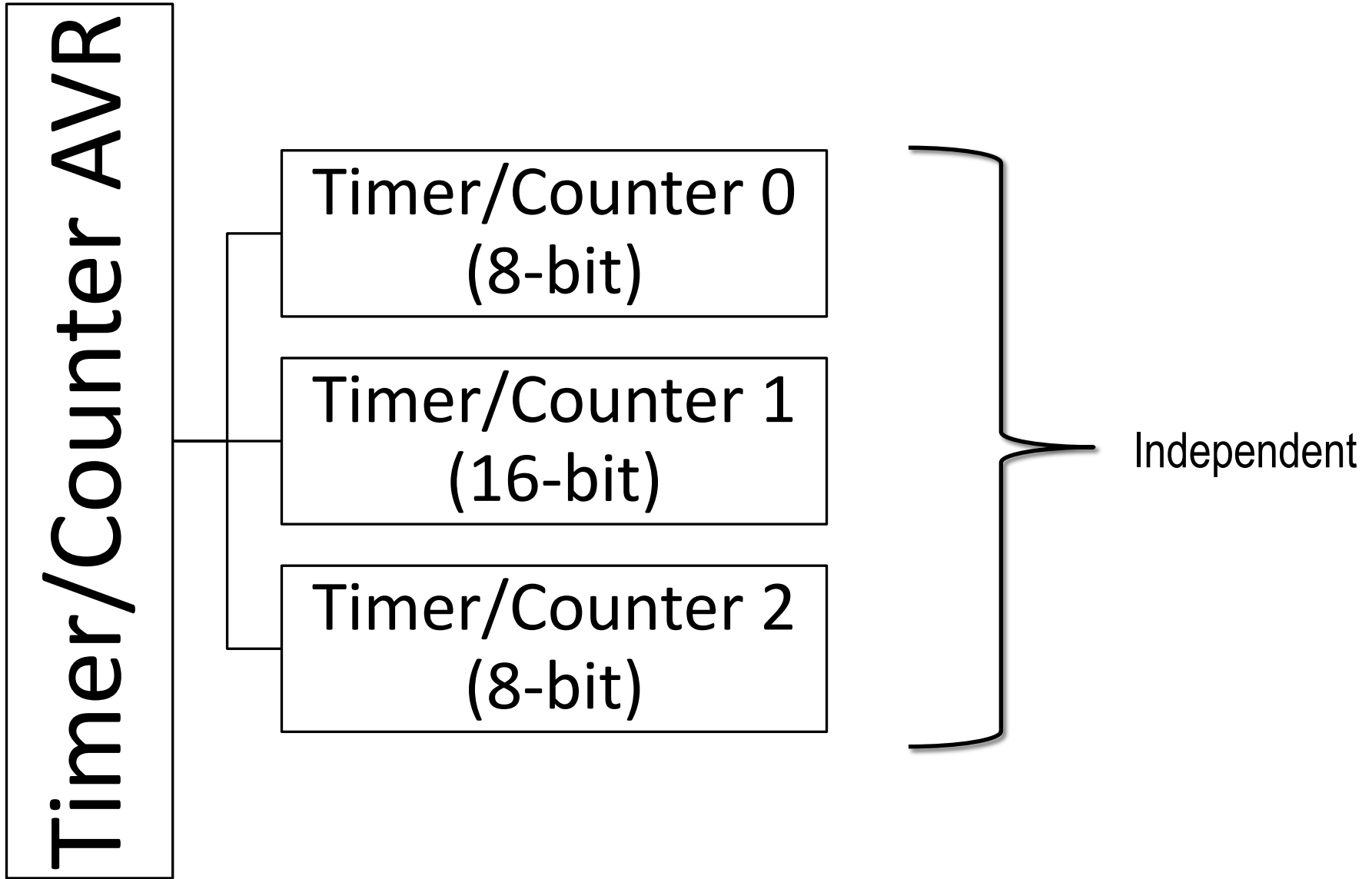
Setelah membahas materi kuliah ini, diharapkan mahasiswa dapat:

1. Memahami blok diagram Timer/Counter serta alur kerja yang berlaku di dalamnya
2. Memahami register-register pada Timer/Counter beserta kegunaannya

Timer/Counter

- Time function (starting program/execution)
- Other uses: timer, PWM, ADC, Oscillator

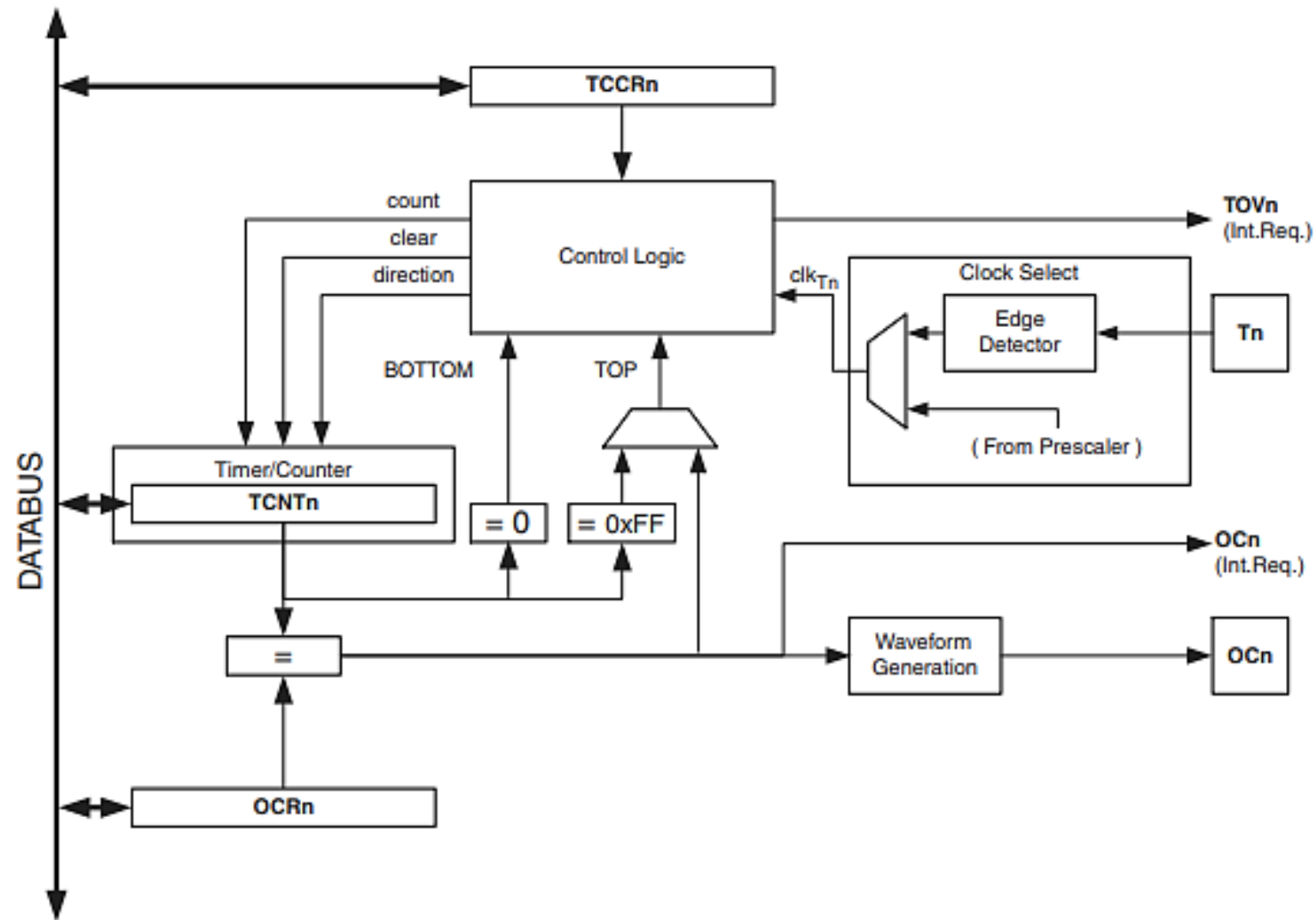




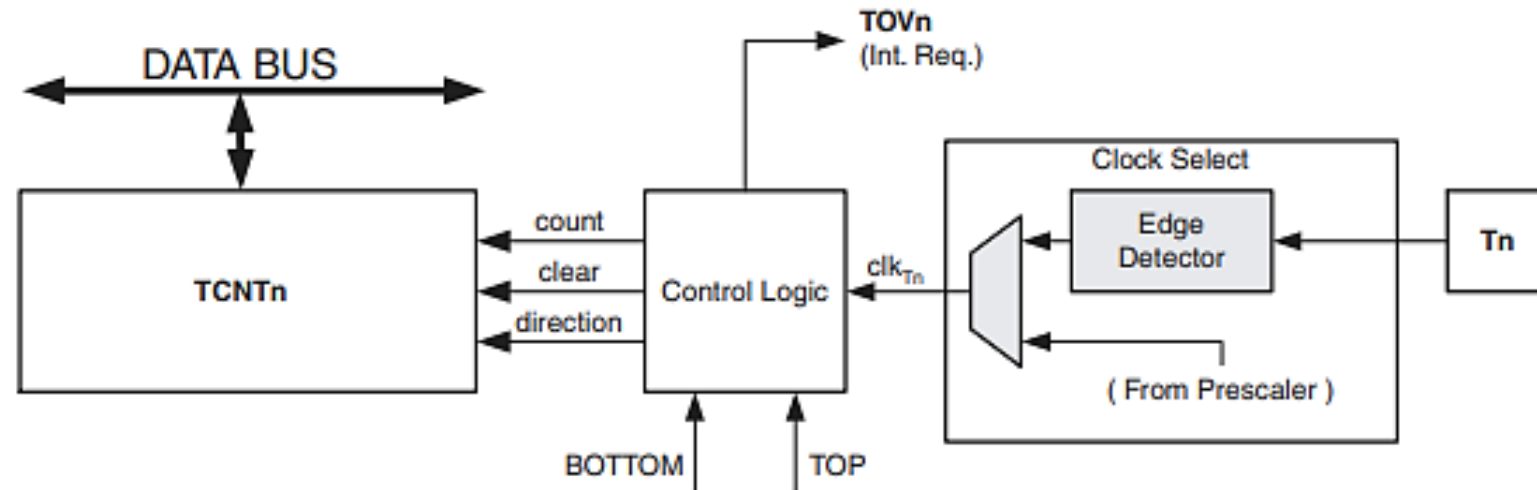
Timer/Counter 0

- Single Compare Unit Counter
- Clear Timer on Compare Match (Auto Reload)
- Phase Correct Pulse Width Modulator (PWM)
- Frequency Generator
- External Event Counter
- 10-bit Clock Prescaler
- Overflow and Compare Match Interrupt Sources (TOV0 and OCF0)

8-bit Timer/Counter Block Diagram



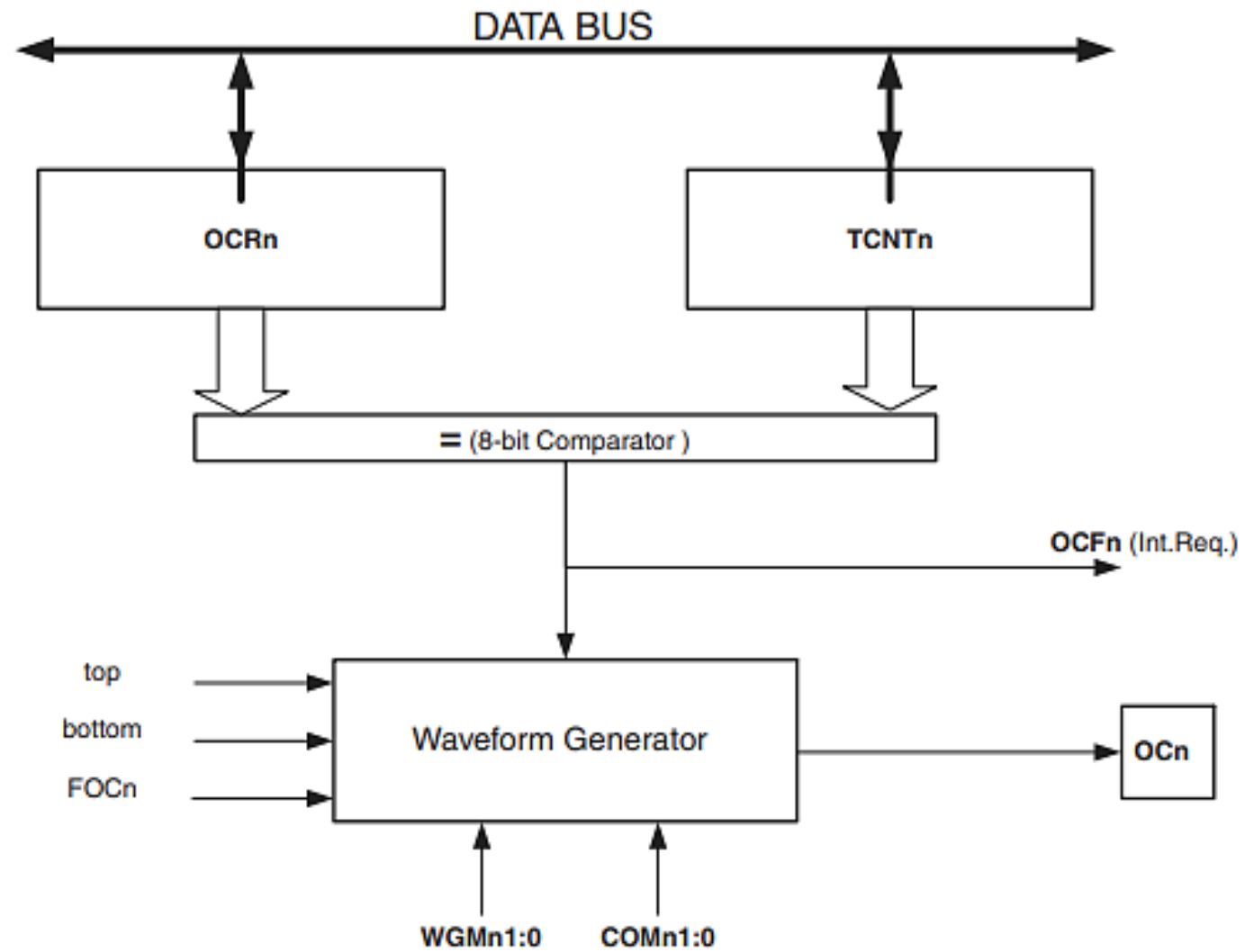
Counter Unit Block Diagram



Signal description (internal signals):

- count** Increment or decrement TCNT0 by 1.
- direction** Select between increment and decrement.
- clear** Clear TCNT0 (set all bits to zero).
- clk_{Tn}** Timer/Counter clock, referred to as clk_{T0} in the following.
- TOP** Signalize that TCNT0 has reached maximum value.
- BOTTOM** Signalize that TCNT0 has reached minimum value (zero).

Output Compare Unit, Block Diagram



Timer/Counter Control Register - TCCR0

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 7 – FOC0: Force Output Compare
- Bit 6,3 – WGM01:0: Waveform Generation Mode
- Bit 5:4 – COM01:0: Compare Match Output Mode
- Bit 2:0 – CS02:0: Clock Select

Clock Select Bit
Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clk _{I/O} /(No prescaling)
0	1	0	clk _{I/O} /8 (From prescaler)
0	1	1	clk _{I/O} /64 (From prescaler)
1	0	0	clk _{I/O} /256 (From prescaler)
1	0	1	clk _{I/O} /1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Timer/Counter Control Register - TCNT0

Bit	7	6	5	4	3	2	1	0	
	<div>TCNT0[7:0]</div>								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Output Compare Register - OCR0

Bit	7	6	5	4	3	2	1	0	
	<div>OCR0[7:0]</div>								OCR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter Interrupt Mask Register - TMSK

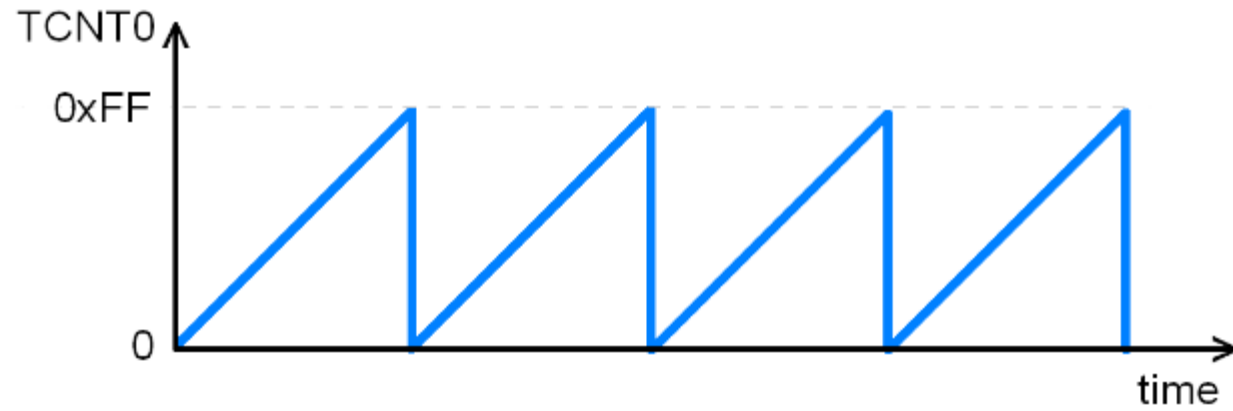
Bit	7	6	5	4	3	2	1	0
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Timer/Counter Interrupt Flag Register - TIFR

[illegible]

Timer0 Overflow

Normal mode: When the counter overflows i.e. goes from 0xFF to 0x00, the TOV0 flag is set



Creating Delay Using Timer0

```
notes * Timer1.c *  
  
#include <avr/io.h>  
  
void T0delay();  
  
int main(void)  
{  
    DDRB = 0xFF;          // PORTB sebagai output  
    while(1)              // Untuk perulangan  
    {  
        PORTB=0x55;  
        T0delay();        // Beri beberapa delay  
        PORTB=0xAA;  
        T0delay();  
    }  
}  
  
void T0delay()  
{  
    TCNT0 = 0x25;          // Load untuk TCNT0  
    TCCR0 = 0x01;          // Timer0, normal mode, no pre-scalar  
  
    while((TIFR&0x01)==0); //Tunggu TOV0 untuk roll over  
    TCCR0 = 0;  
    TIFR = 0x1;            /* Clear flag pada TOV0  
}
```

Counting Delay (Fosc= 8MHz)

$$T = \frac{1}{F_{osc}} = \frac{1}{8 \times 10^6} = 0,125 \times 10^{-6} = 0,125 \mu s$$

TCNT = 0x25

0xFF – 0x25 = 0xDA (218 desimal)

Tambahkan 1 siklus lagi untuk membangkitkan flag
TOV0 = 219.

Total delay = 219 × 0,125 μs = **27,375 μs**

Timer Input Capture Mode in AVR ATmega16/ATmega32

The input capture function is used in many applications such as:

- Pulse width measurement
- Period measurement
- Capturing the time of an event

In AVR ATmega32, Timer1 can be used as input capture to detect and measure events happening outside the microcontroller.

Upon detection of a defined event i.e. rising edge or falling edge on ICP pin (PORTD.6), the TCNT1 (Timer/Counter register) value is loaded into the OCR1 (input capture) register and the ICF1 flag will get set.



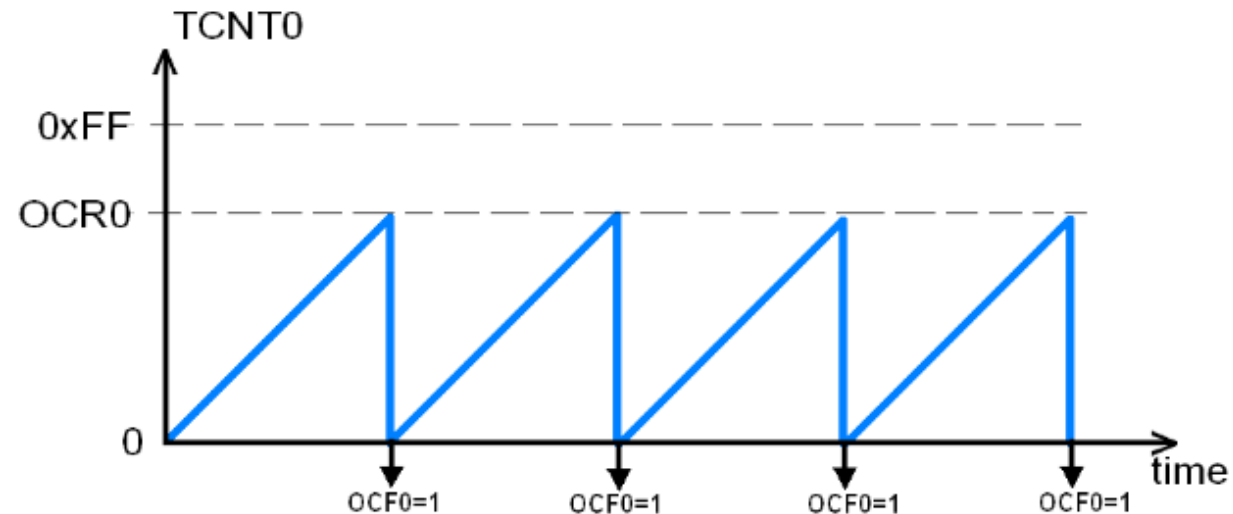
Clear Timer on Compare Match (CTC mode) in AVR ATmega16/ATmega32

Generally, compare model is used for generating periodic events or for generating waveforms.

In compare mode, there is one compare register, where we can set the value to compare with the Timer/Counter register value. Once the compare value matches with the timer/counter register value, a compare match occurs. The compare match event can be used for waveform generation.

In ATmega16/32, the Timer counts up until the value of TCNT0 (Timer/Counter register) register becomes equal to the content of OCR0 (Compare register). As soon as TCNT0 becomes equal to the OCR0, a compare match occurs, and then the timer will get cleared and the OCF0 flag will get set.

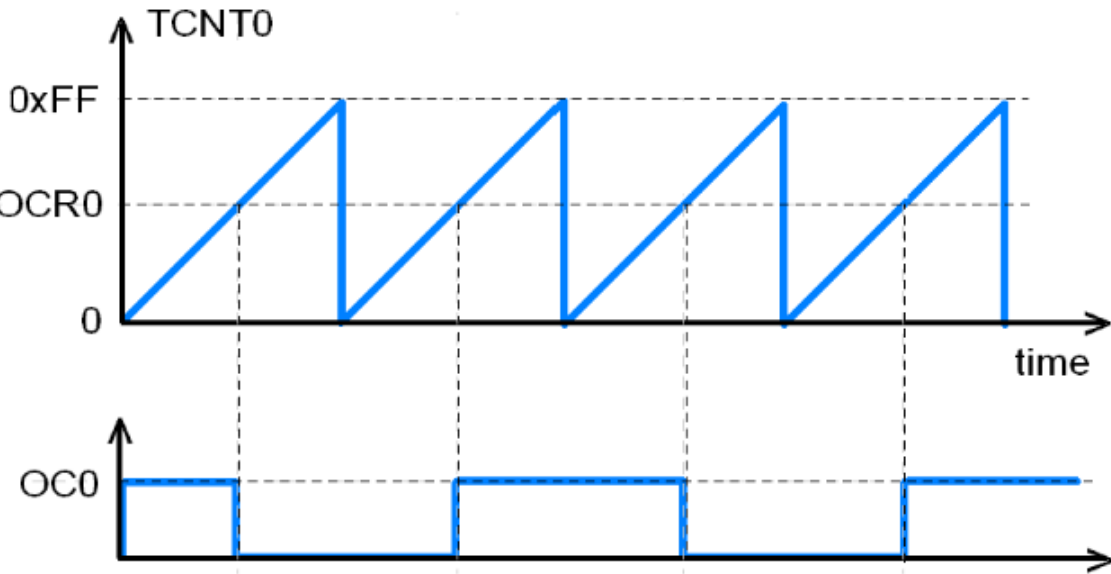
OCF0 flag is located in the TIFR register.



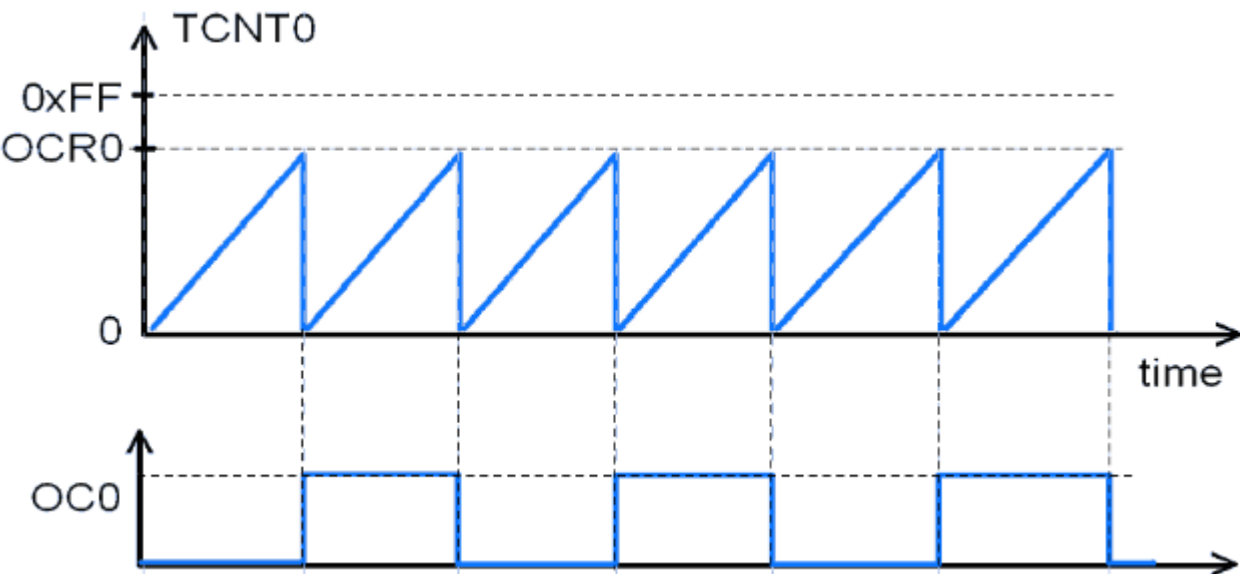
Clear Timer on Compare Match (CTC mode) in AVR ATmega16/ATmega32

```
#include "avr/io.h"
int main ( )
{
    DDRB = DDRB | (1<<3);
    TCCR0 = 0x11;    // normal mode, clk- no pre-scaling
    OCR0 = 100; // nilai compare compare value */
    while (1);
    return 0;
}
```

Normal Mode Vs CTC Mode



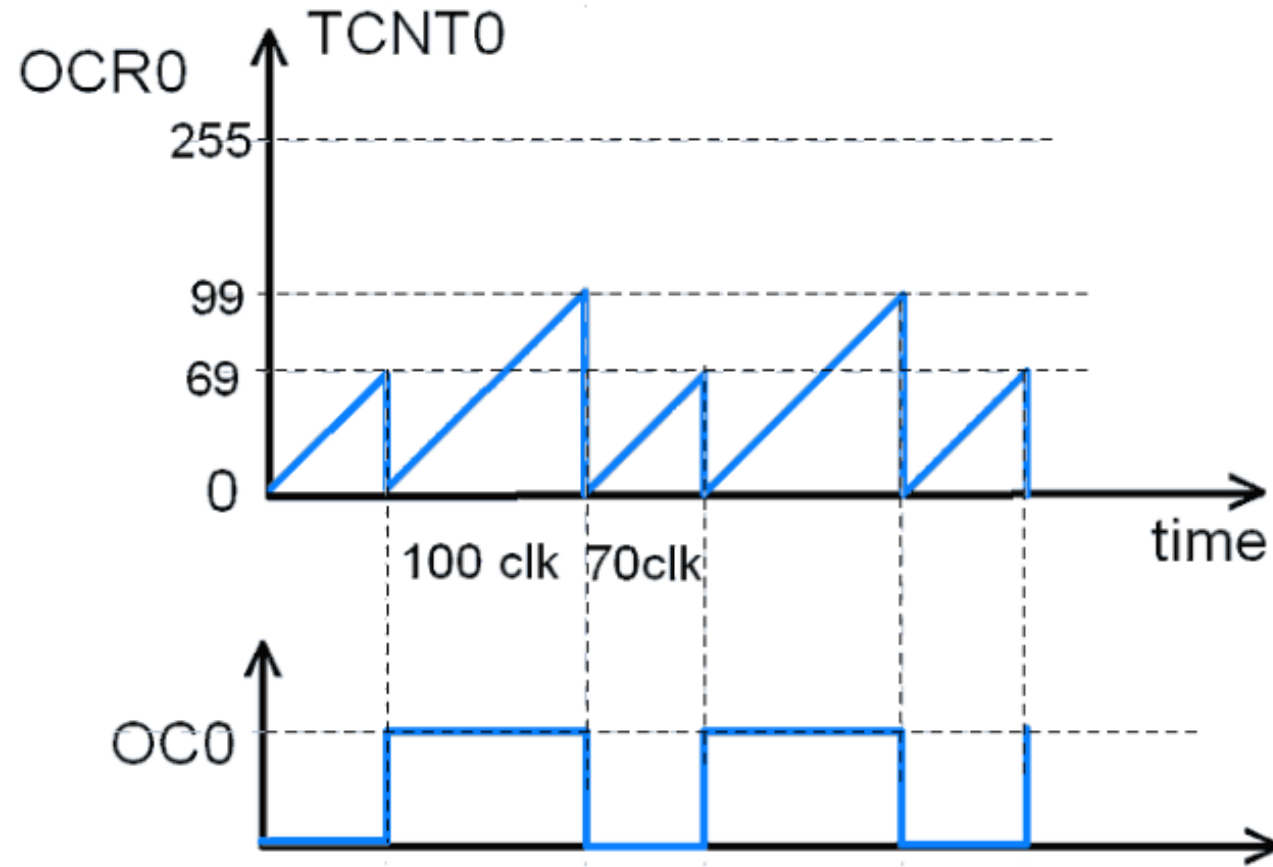
Waveform Generation Using Normal Mode



Waveform Generation Using CTC Mode

CTC Mode

We can change the value of OCR0 in runtime, to generate different pulses



SEKIAN DAN TERIMA KASIH

PENGANTAR MIKROKONTROLER

Pertemuan 6

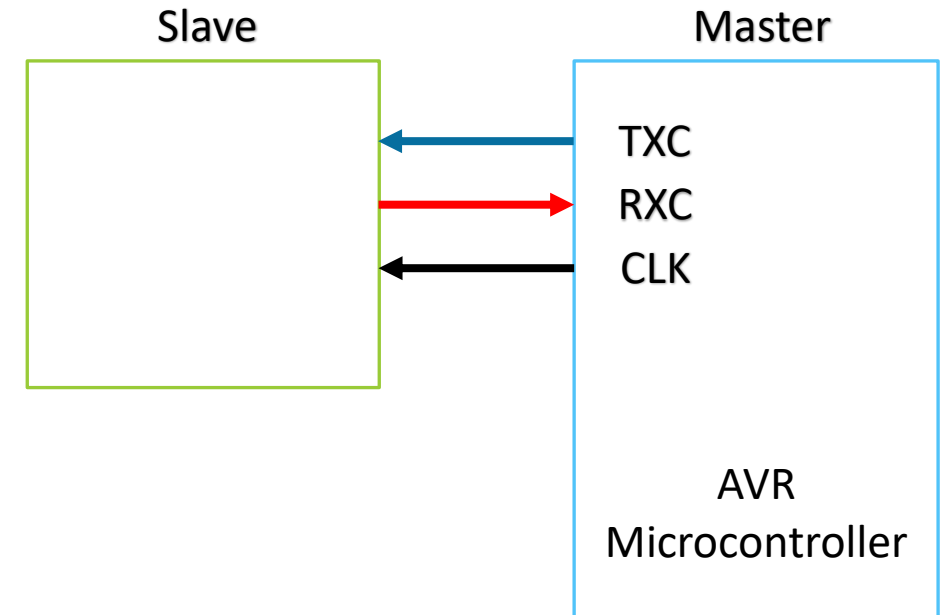
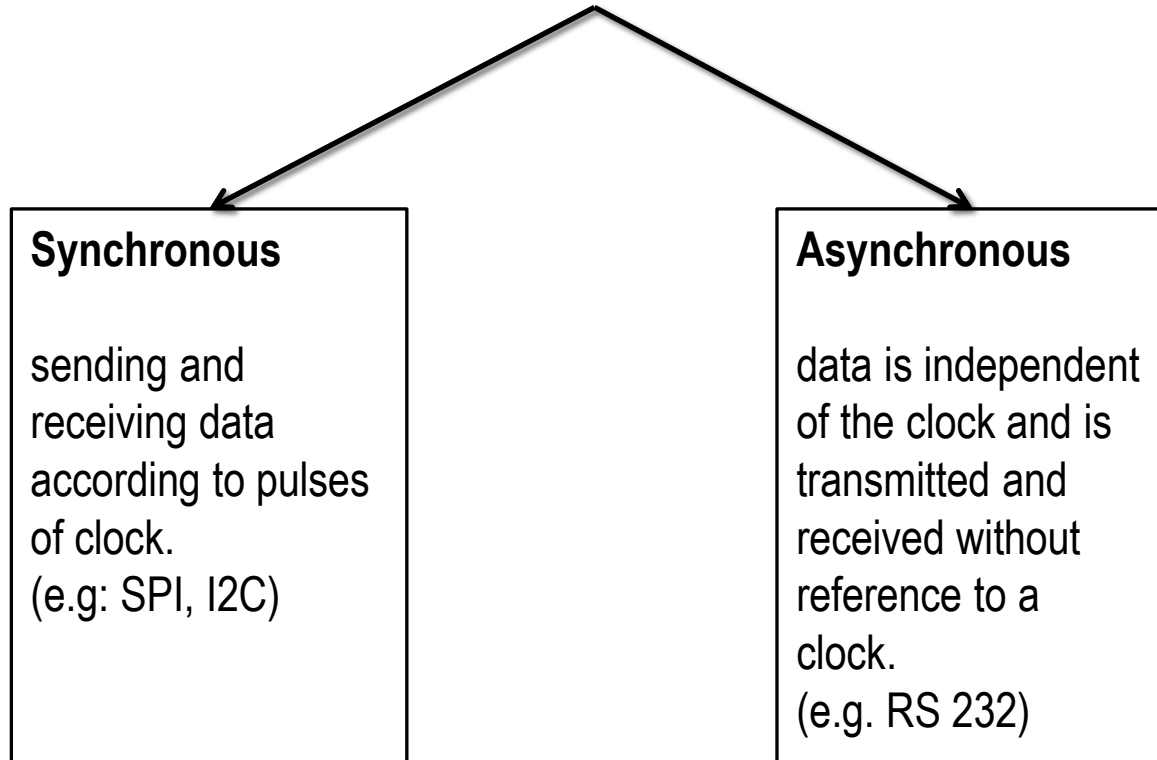
Ahmad Zarkasi



MATERI BAHASAN

USART/UART in AVR ATmega16/ATmega32

USART (Universal Synchronous Asynchronous Receiver Transmitter)



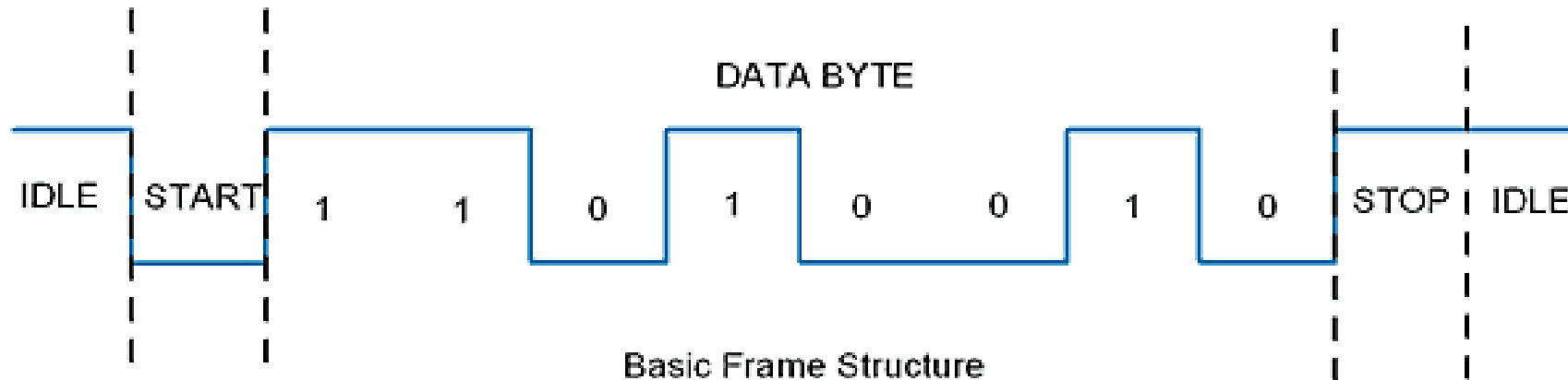
Introduction

AVR ATmega has flexible USART, which can be used for serial communication with other devices like computers, serial GSM, GPS modules, etc.

Before beginning with AVR USART, we will through the basic of serial communication.

Serial data framing

While sending/receiving data, some bits are added for the purpose of knowing the beginning/ending of data, etc. commonly used structure is: 8 data bits, 1 start bit (logic 0), and 1 stop bit (logic 1), as shown:



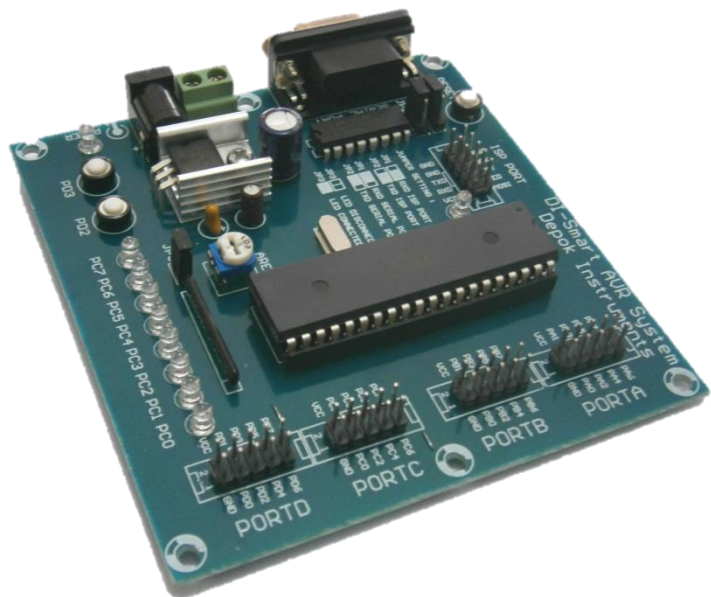
Speed (Baud Rate)

As we know the bit rate is “Number of bits per second (bps)”, also known as baud rate in binary system. Normally this defines how fast the serial line is. There are some standard baud rates defined e.g. 1200, 2400, 4800, 19200, 115200 bps, etc. Normally 9600 bps is used where speed is not critical issue.

Wires and Hardware Connection

Normally in USART, we only need Tx (Transmit), Rx (Receive), and GND wires.

- AVR Atmega USART has a TTL voltage level which is 0V for logic 0 and 5V for logic 1.
- In computers and most of the old devices, RS232 protocol is used for serial communication i.e. +3V to +25V for logic zero and -3V to -25V for logic 1.



Tx



Rx

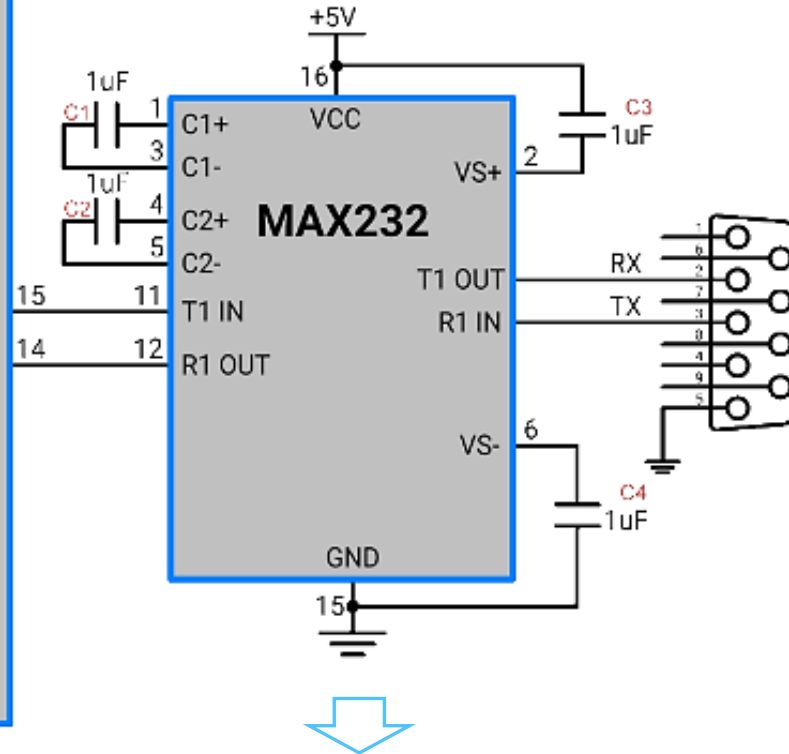


ATmega16/32 Serial Interface

TTL	
LOGIC 0	0 V
LOGIC 1	5 V



RS232	
LOGIC 0	+3 to +25 V
LOGIC 1	-3 to -25 V



Converter

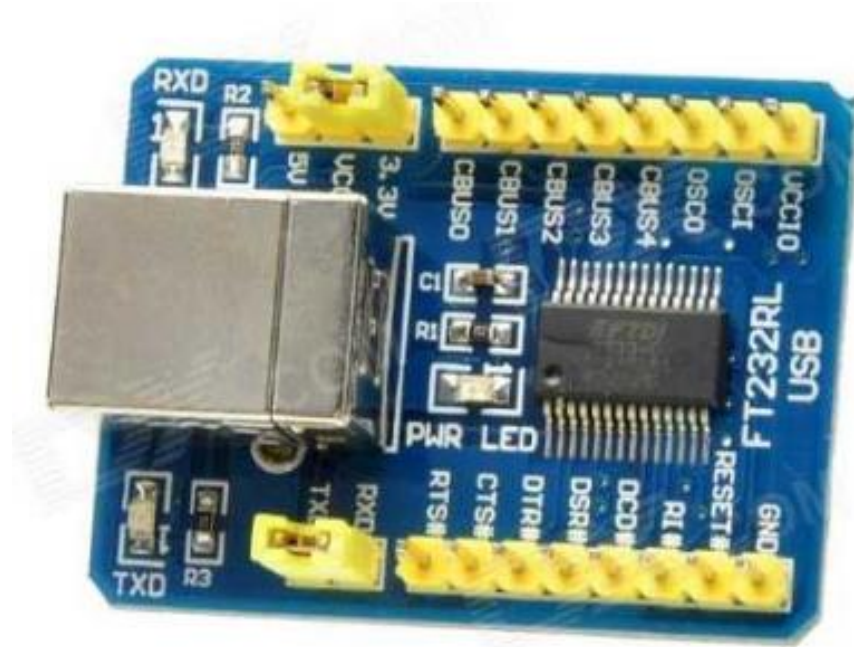


RS 232
to USB



Komputer

With a new PC and laptops, there is no RS232 protocol and DB9 connector. We have to use serial to USB connector. There are various serial to USB connectors available e.g. CP2102, FT232RL, CH340, etc.

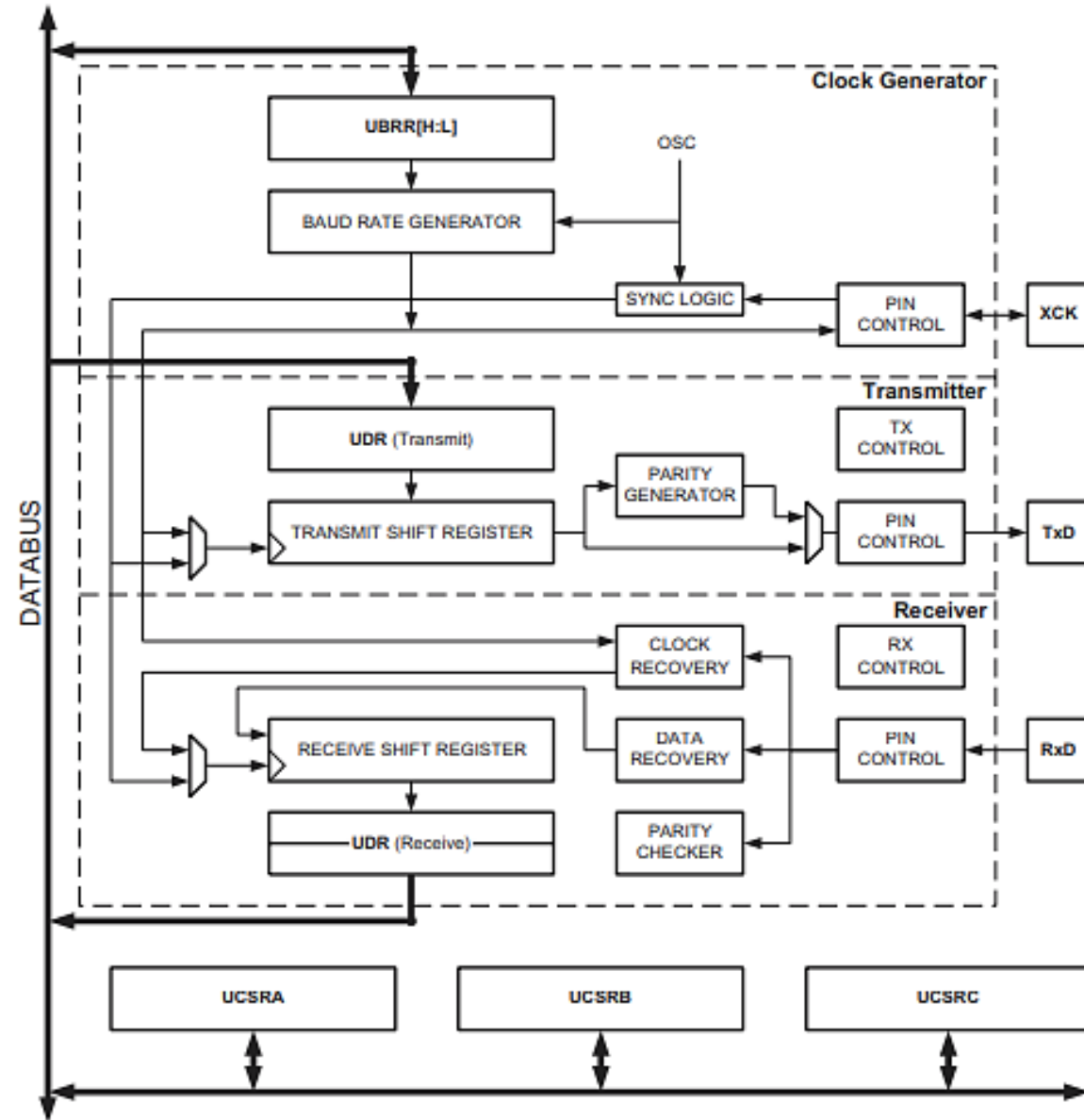


Serial to USB Converter

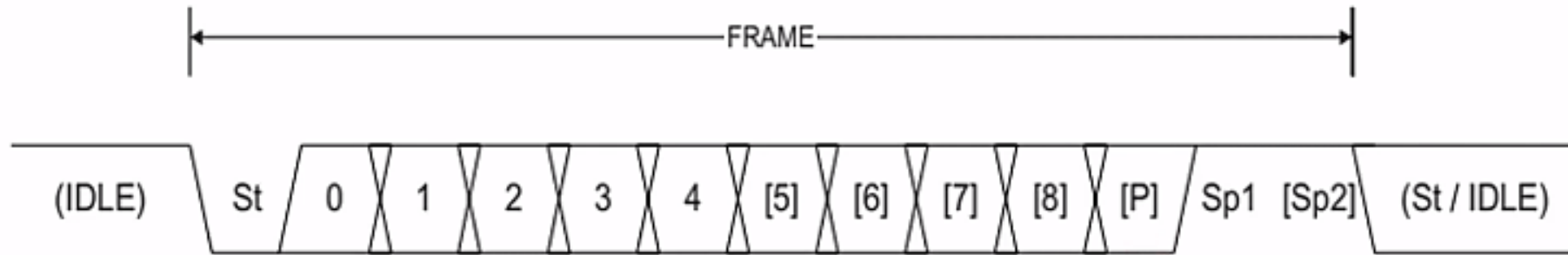
Features

- Full Duplex Operation (Independent Serial Receive and Transmit Registers)
- Asynchronous or Synchronous Operation
- Master or Slave Clocked Synchronous Operation
- High-Resolution Baud Rate Generator
- Supports Serial Frames with 5, 6, 7, 8, or 9 Data Bits and 1 or 2 Stop Bits
- Odd or Even Parity Generation and Parity Check Supported by Hardware (Data Validation)
- Data Over Run Detection
- Framing Error Detection
- Noise Filtering Includes False Start Bit Detection and Digital Low Pass Filter
- Three Separate Interrupts on TX Complete, TX Data Register Empty, and RX Complete
- Etc.

USART Block Diagram



Data Frame



- St : Start Bit
- n : Data Bit (0-8)
- Sp : Stop Bit
- P : Parity Bit

Registers

- UCSRA
- UCSRB
- UCSRC
- USRRH
- USRRL
- UDR

UBRR0L and UBRR0H – USART Baud Rate Registers

Bit	15	14	13	12	11	10	9	8	
	URSEL	–	–	–	UBRR (11:8)				UBRRH
	UBRR (7:0)								UBRRL
	7	6	5	7	3	2	1	0	

- Bit 15 – URSEL : Register Select (low:UBRH), (high:UCSRC)
- Bit 14-12 – Reserve Bit
- Bit 11:10 – UBRR1:0 (Baudrate Setting)

$$UBRR = (f_{clock}/16*Baud) - 1$$

UCSR0A – USART Control and Status Register A

Bit	7	6	5	4	3	2	1	0
	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Read/Write	R	R/W	R	R	R	R	R	R
Initial Value	0	0	1	0	0	0	0	0

- **Bit 7 – RXC: USART Receive Complete** ✓ = 1, jika data siap dibaca

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the receiver is disabled, the receive buffer will be flushed and consequently the RXC bit will become zero. The RXC flag can be used to generate a Receive Complete interrupt (see description of the RXCIE bit).

- **Bit 6 – TXC: USART Transmit Complete** ✓ = 1, jika semua data terkirim

This flag bit is set when the entire frame in the transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR). The TXC flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC flag can generate a Transmit Complete interrupt (see description of the TXCIE bit).

- **Bit 5 – UDRE: USART Data Register Empty** ✓ = 1, jika datanya kosong

The UDRE flag indicates if the transmit buffer (UDR) is ready to receive new data. If UDRE is one, the buffer is empty, and therefore ready to be written. The UDRE flag can generate a Data Register empty Interrupt (see description of the UDRIE bit).

UDRE is set after a reset to indicate that the transmitter is ready.

- **Bit 4 – FE: Frame Error**

This bit is set if the next character in the receive buffer had a Frame Error when received. i.e., when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDR) is read. The FE bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSRA.

- **Bit 3 – DOR: Data OverRun**

This bit is set if a Data OverRun condition is detected. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the receive Shift Register, and a new start bit is detected. This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

- **Bit 2 – PE: Parity Error**

This bit is set if the next character in the receive buffer had a Parity Error when received and the parity checking was enabled at that point (UPM1 = 1). This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

- **Bit 1 – U2X: Double the USART Transmission Speed**

This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.

Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

- **Bit 0 – MPCM: Multi-processor Communication Mode**

This bit enables the Multi-processor Communication mode. When the MPCM bit is written to one, all the incoming frames received by the USART receiver that do not contain address information will be ignored. The transmitter is unaffected by the MPCM setting. For more detailed information see "Multi-processor Communication Mode" on page 154.

UCSR0B – USART Control and Status Register B

Bit	7	6	5	4	3	2	1	0
	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – RXCIE: RX Complete Interrupt Enable** ✓

Writing this bit to one enables interrupt on the RXC flag. A USART Receive Complete Interrupt will be generated only if the RXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC bit in UCSRA is set.

- **Bit 6 – TXCIE: TX Complete Interrupt Enable** ✓

Writing this bit to one enables interrupt on the TXC flag. A USART Transmit Complete Interrupt will be generated only if the TXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC bit in UCSRA is set.

- **Bit 5 – UDRIE: USART Data Register Empty Interrupt Enable**

Writing this bit to one enables interrupt on the UDRE flag. A Data Register Empty Interrupt will be generated only if the UDRIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE bit in UCSRA is set.

- **Bit 4 – RXEN: Receiver Enable** ✓

Writing this bit to one enables the USART Receiver. The Receiver will override normal port operation for the RxD pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FE, DOR, and PE flags.

- **Bit 3 – TXEN: Transmitter Enable** ✓

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD pin when enabled. The disabling of the Transmitter (writing TXEN to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the transmit Shift Register and transmit Buffer Register do not contain data to be transmitted. When disabled, the transmitter will no longer override the TxD port.

- **Bit 2 – UCSZ2: Character Size**

The UCSZ2 bits combined with the UCSZ1:0 bit in UCSRC sets the number of data bits (Character Size) in a frame the receiver and transmitter use.

- **Bit 1 – RXB8: Receive Data Bit 8**

RXB8 is the ninth data bit of the received character when operating with serial frames with nine data bits. Must be read before reading the low bits from UDR.

- **Bit 0 – TXB8: Transmit Data Bit 8**

TXB8 is the ninth data bit in the character to be transmitted when operating with serial frames with nine data bits. Must be written before writing the low bits to UDR.

UCSR0C – USART Control and Status Register C

Bit	7	6	5	4	3	2	1	0
	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ0	UCPOLO
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	1	1	0

- **Bit 7 – URSEL: Register Select** ✓ 0 → UBRRH, 1 → UCSRC
This bit selects between accessing the UCSRC or the UBRRH Register. It is read as one when reading UCSRC. The URSEL must be one when writing the UCSRC.
- **Bit 6 – UMSEL: USART Mode Select** ✓ 0 → asynchronous, 1 → synchronous
This bit selects between Asynchronous and Synchronous mode of operation.
- **Bit 5:4 – UPM1:0: Parity Mode** ✓
These bits enable and set type of parity generation and check. If enabled, the transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data and compare it to the UPM0 setting. If a mismatch is detected, the PE flag in UCSRA will be set.
- **Bit 3 – USBS: Stop Bit Select** ✓
This bit selects the number of Stop Bits to be inserted by the Transmitter. The Receiver ignores this setting.
- **Bit 2:1 – UCSZ1:0: Character Size** ✓
Menentukan banyak bit dalam 1 frame
The UCSZ1:0 bits combined with the UCSZ2 bit in UCSRB sets the number of data bits (Character Size) in a frame the Receiver and Transmitter use.

UDR0 – USART I/O Data Register

Bit	7	6	5	4	3	2	1	0
UDR (Read)	RXB7	RXB6	RXB5	RXB4	RXB3	RXB2	RXB1	RXB0
UDR (Write)	TXB7	TXB6	TXB5	TXB4	TXB3	TXB2	TXB1	TXB0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- All data received by the receiver and data sent by the transmitter will be accommodated by the UDR
- To read the data received by the receiver can be done by reading the contents of this UDR.
- To send data through the transmitter it is enough to provide a UDR value.

USART Initialization

```
void USART_Init( unsigned int baud )
{
    /* Set baud rate */
    UBRRH = (unsigned char) (baud>>8);
    UBRRL = (unsigned char)baud;
    /* Enable receiver and transmitter */
    UCSRB = (1<<RXEN) | (1<<TXEN);
    /* Set frame format: 8data, 2stop bit */
    UCSRC = (1<<URSEL) | (1<<USBS) | (3<<UCSZ0);
}
```

The background is a dark, textured surface covered in numerous small, colorful splatters and streaks in shades of blue, green, and yellow. A prominent, bright, horizontal light streak or lens flare effect runs across the middle of the image, transitioning from a deep blue on the left to a bright yellow and white on the right.

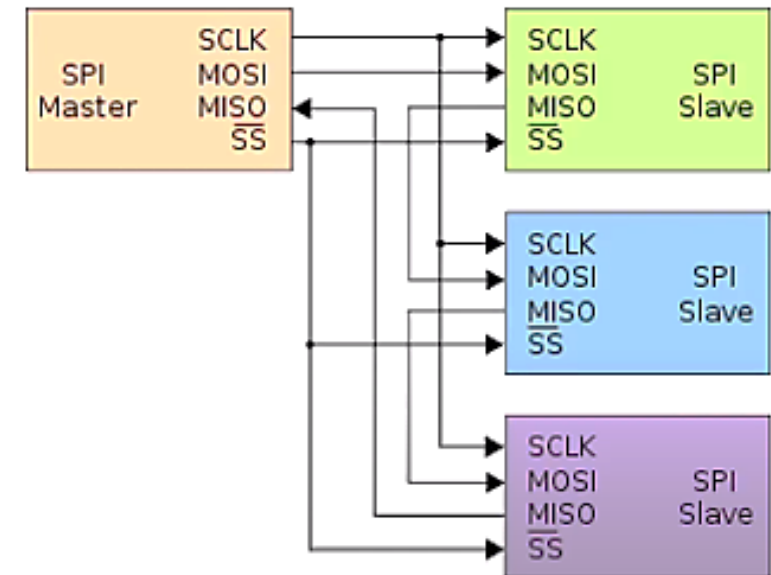
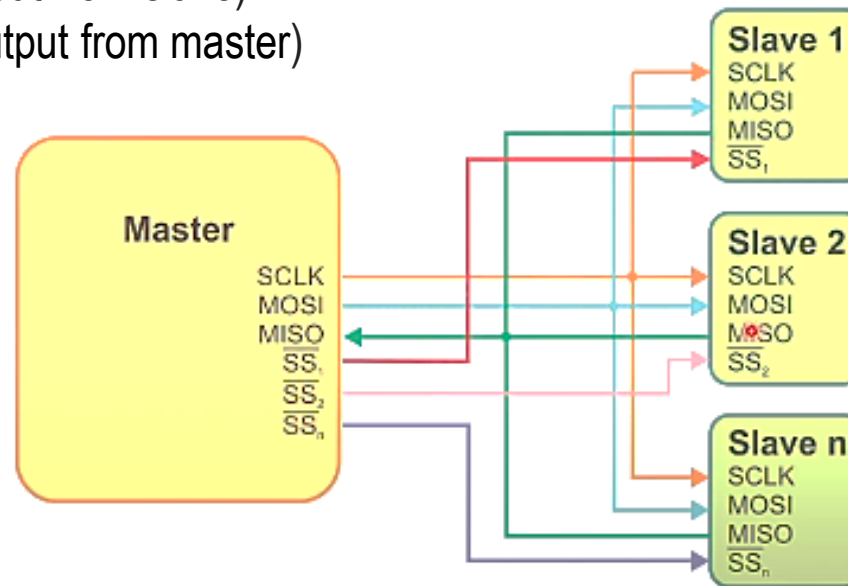
MATERI BAHASAN

SPI (Serial Peripheral Interface)

SPI: The Serial Peripheral Interface (SPI) is a synchronous serial communication interface specification used for short-distance communication, primarily in embedded systems. SPI is an interface bus commonly used to send data between microcontrollers and small peripherals such as shift registers, sensors, and SD cards. It uses separate clock and data lines, along with a select line to choose the device you wish to talk to.

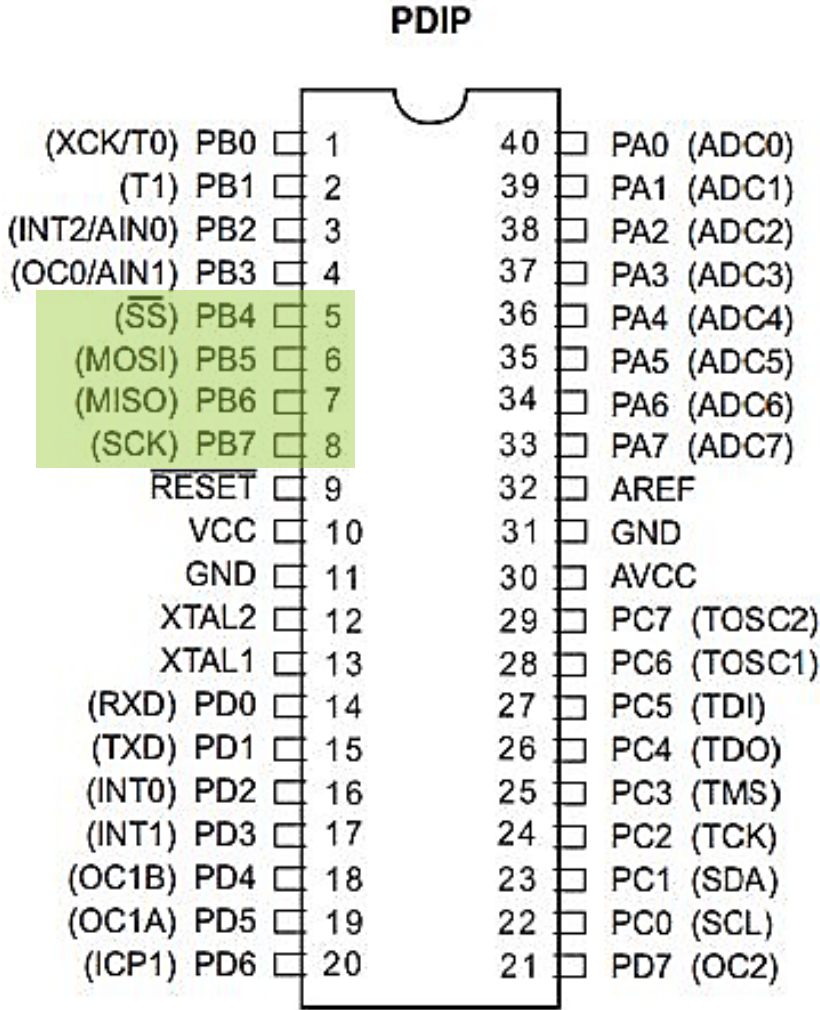
The SPI bus specifies four logic signals:

- SCLK: Serial Clock (output from master)
- MOSI: Master Out Slave In (data output from master)
- MISO: Master In Slave Out (data output from slave)
- SS: Slave Select (often active low, output from master)



SPI Pin

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input
MISO	Input	User Defined
SCK	User Defined	Input
\overline{SS}	User Defined	Input



SPI Control Register - SPCR

Bit	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 7 – SPIE: SPI Interrupt Enable ✓

This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and the if the global interrupt enable bit in SREG is set.

- Bit 6 – SPE: SPI Enable ✓

When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

- Bit 5 – DORD: Data Order

When the DORD bit is written to one, the LSB of the data word is transmitted first.

When the DORD bit is written to zero, the MSB of the data word is transmitted first.

- Bit 4 – MSTR: Master/Slave Select ✓

This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic zero. If SS is configured as an input and is driven low while MSTR is set, MSTR will be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI Master mode.

- Bit 3 – CPOL: Clock Polarity

- Bit 2 – CPHA: Clock Phase

- Bits 1, 0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0

SPI Status Register - SPSR

Bit	7	6	5	4	3	2	1	0	
	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIF: SPI Interrupt Flag** ✓

When a serial transfer is complete, the SPIF flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. If \overline{SS} is an input and is driven low when the SPI is in Master mode, this will also set the SPIF flag.

- **Bit 6 – WCOL: Write COLLision flag**

The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared by first reading the SPI Status Register with WCOL set, and then accessing the SPI Data Register.

- **Bit 5..1 – Res: Reserved Bits**

These bits are reserved bits in the ATmega16 and will always read as zero.

- **Bit 0 – SPI2X: Double SPI Speed Bit**

When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (see Table 58). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at $f_{osc}/4$ or lower.

SPI Data Register - SPDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

The SPI Data Register is a read/write register used for data transfer between the register file and the SPI Shift Register. Writing to the register initiates data transmission. Reading the register causes the Shift Register Receive buffer to be read.

```
void SPI_SlaveInit(void)
{
    /* Set MISO output, all others input */
    DDR_SPI = (1<<DD_MISO);
    /* Enable SPI */
    SPCR = (1<<SPE);
}

char SPI_SlaveReceive(void)
{
    /* Wait for reception complete */
    while(!(SPSR & (1<<SPIF)))
        ;
    /* Return data register */
    return SPDR;
}
```

SEKIAN DAN TERIMA KASIH

PENGANTAR MIKROKONTROLER

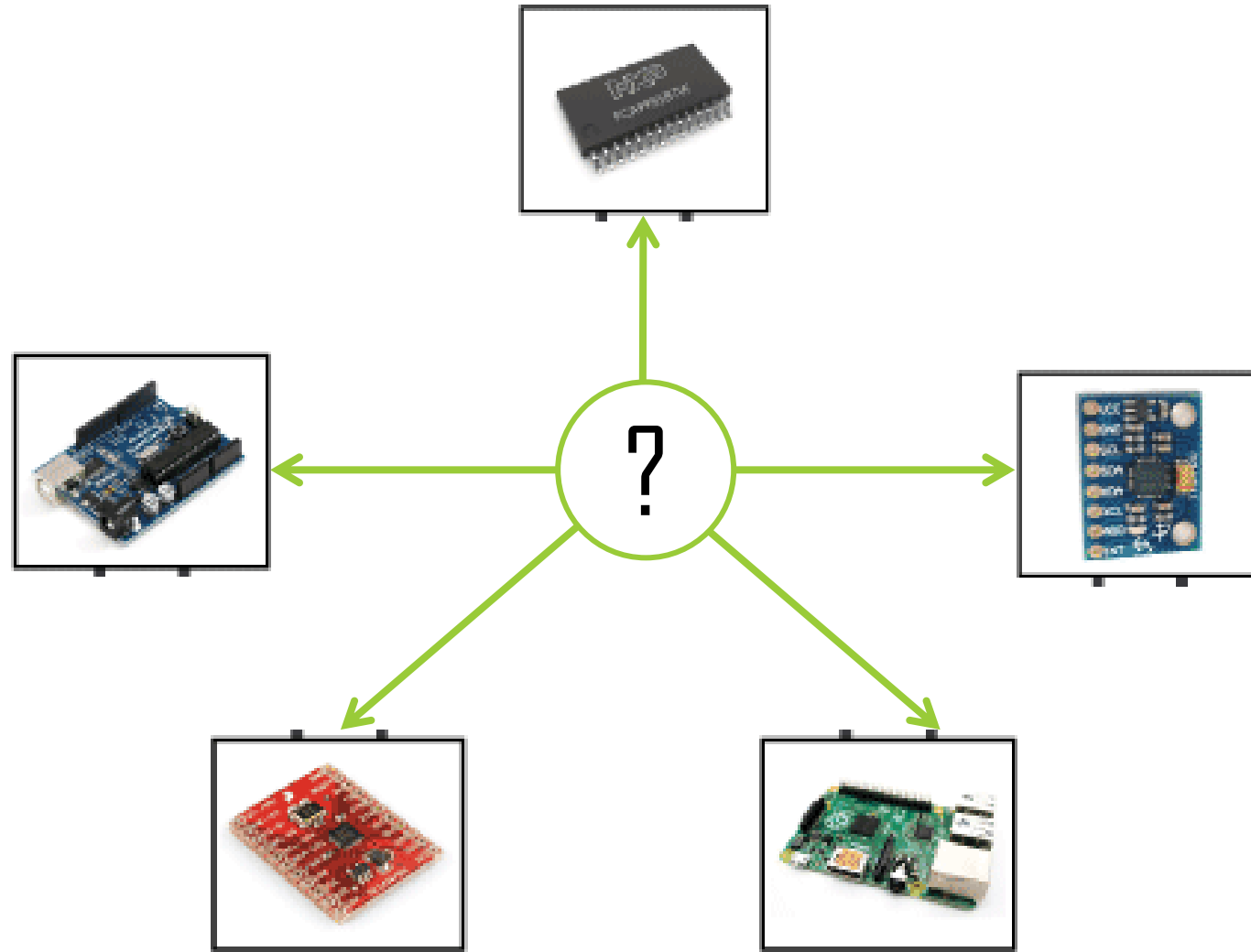
Pertemuan 7

Ahmad Zarkasi



MATERI BAHASAN

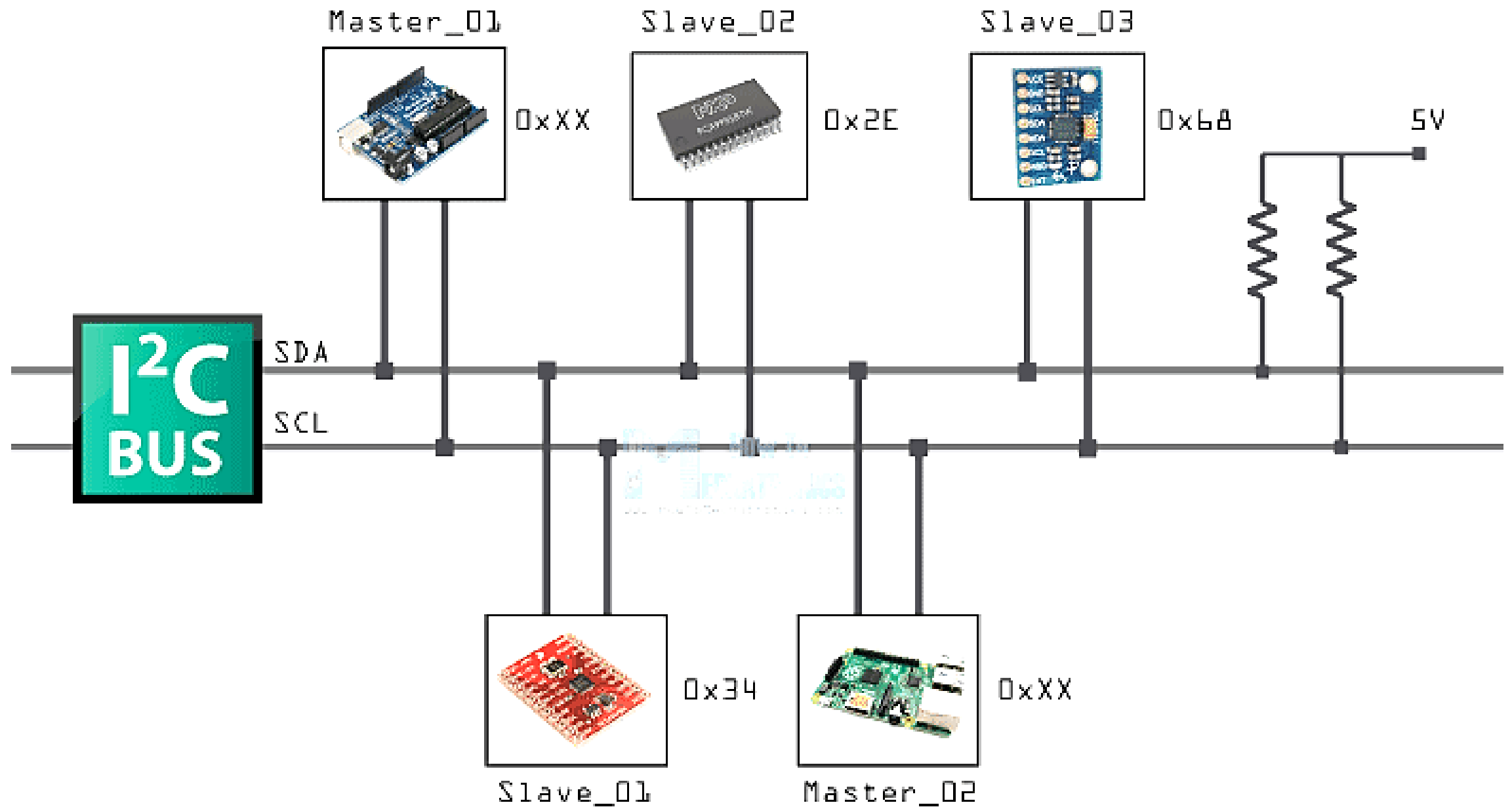
TWO WIRE INTERFACE (TWI)



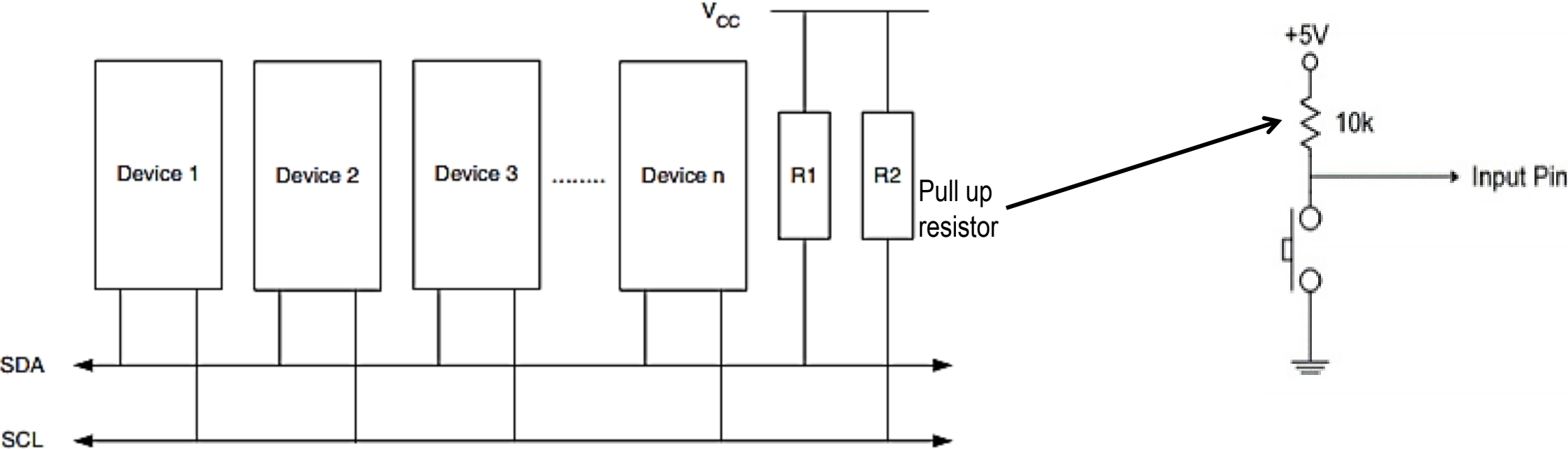
Basics

The **I²C (Inter-Integrated Circuit)** protocol, referred to as *I-squared-C*, *I-two-C*, or *IIC*) is two wire serial communication protocol for connecting low speed peripherals to a microcontroller or computer motherboard. I²C is ideally suited for typical microcontroller applications.

The TWI protocol allows the system designer to interconnect up to 128 different devices using only two bi-directional bus lines, one for clock (SCL) and one for data (SDA). The only external hardware needed to implement the bus is a single pull-up resistor for each of the bus lines.



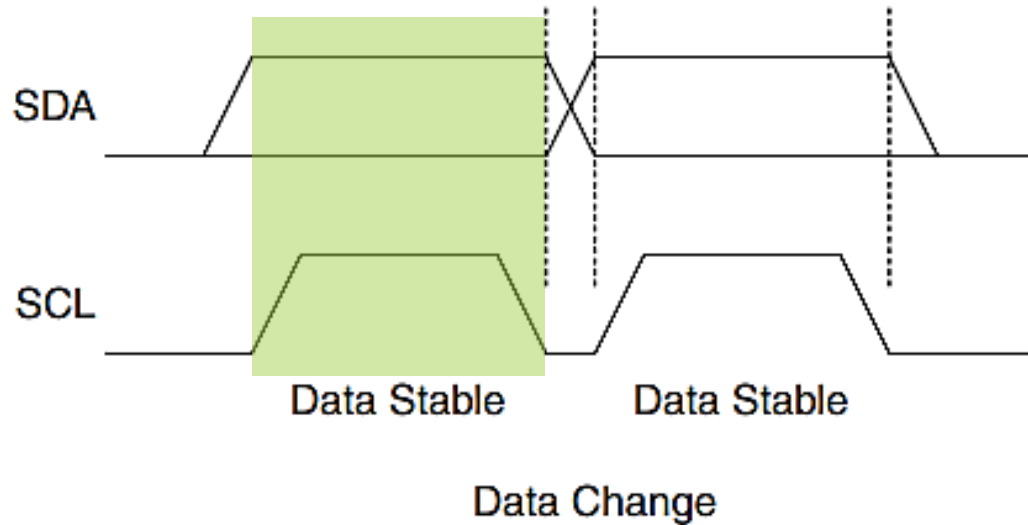
TWI Bus Interconnection



TWI Terminology

Term	Description
Master	The device that initiates and terminates a transmission. The master also generates the SCL clock.
Slave	The device addressed by a master.
Transmitter	The device placing data on the bus.
Receiver	The device reading data from the bus.

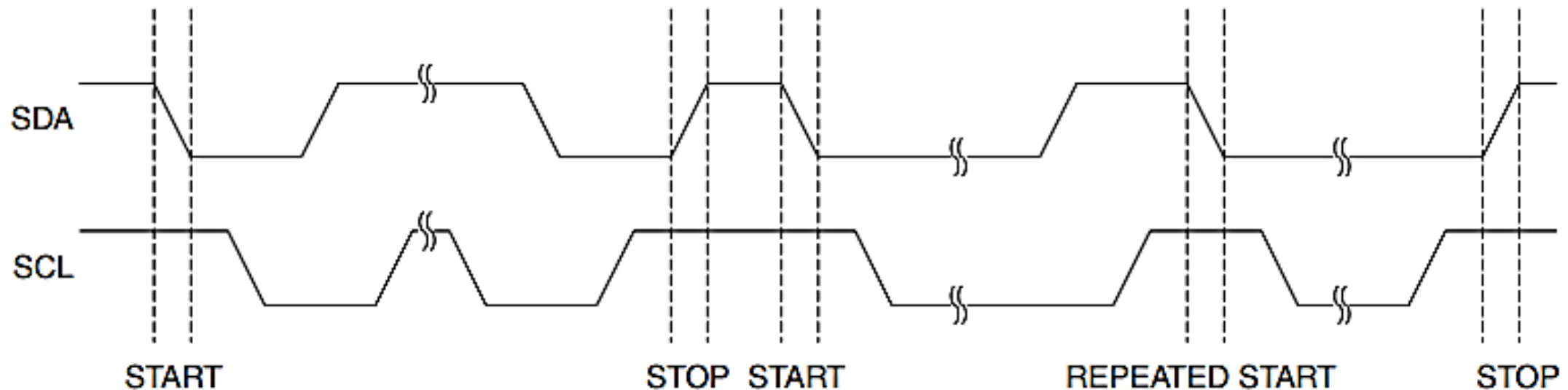
Data Validity



$SDA = 0 \text{ or } 1 \ \& \ SCL = 1 \rightarrow \text{Valid : 1 bit data}$

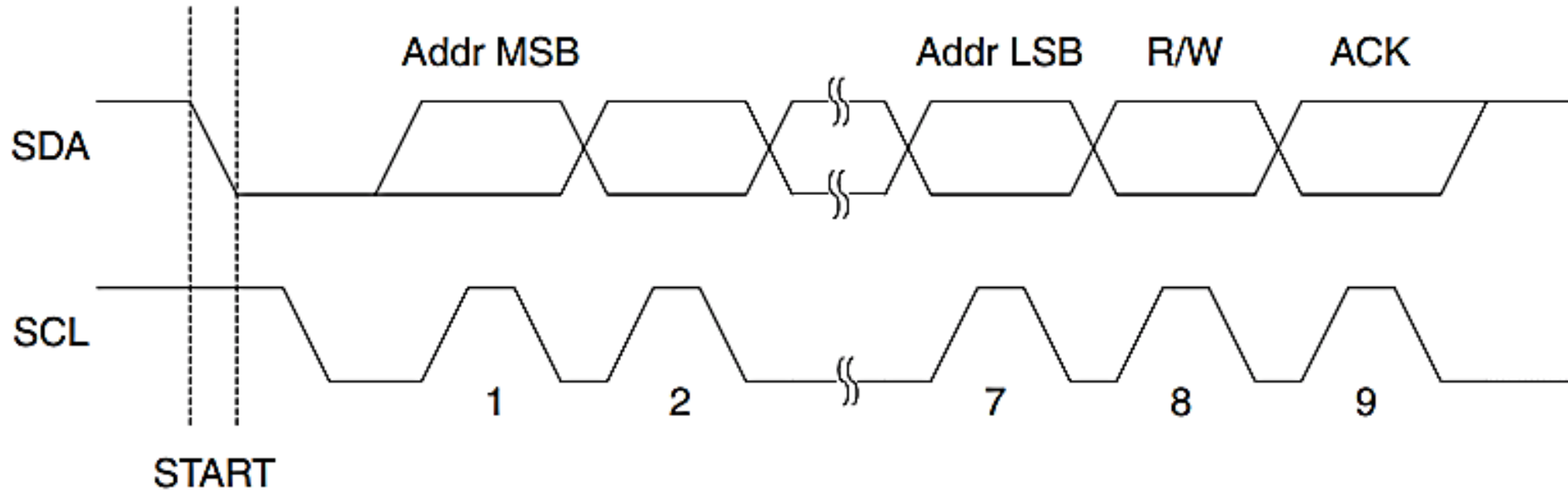
START, REPEATED START, and STOP Conditions

The master initiates and terminates a data transmission. The transmission is initiated when the master issues a START condition on the bus, and it is terminated when the master issues a STOP condition. Between a START and a STOP condition, the bus is considered busy, and no other master should try to seize control of the bus. A special case occurs when a new START condition is issued between a START and STOP condition. This is referred to as a REPEATED START condition, and is used when the master wishes to initiate a new transfer without releasing control of the bus. After a REPEATED START, the bus is considered busy until the next STOP.



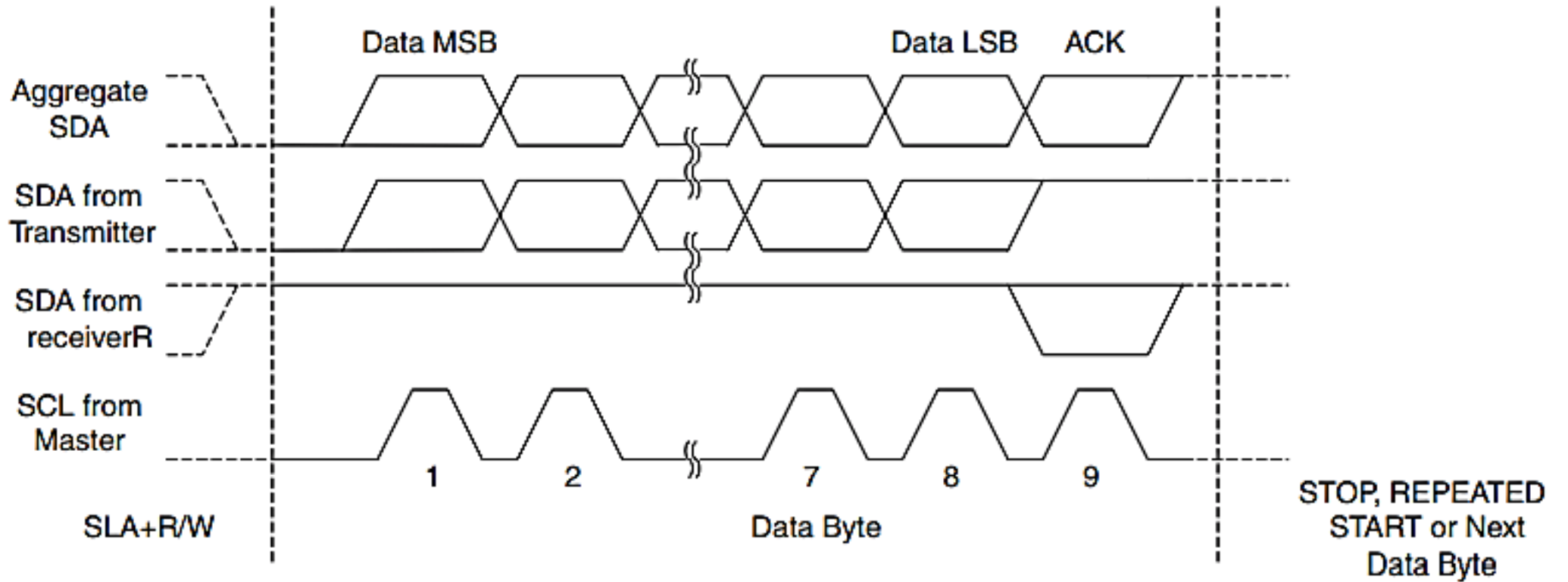
Address Packet Format

All address packets transmitted on the TWI bus are nine bits long, consisting of seven address bits, one READ/WRITE control bit and an acknowledge bit. If the READ/WRITE bit is set, a read operation is to be performed, otherwise a write operation should be performed. When a slave recognizes that it is being addressed, it should acknowledge by pulling SDA low in the ninth SCL (ACK) cycle.

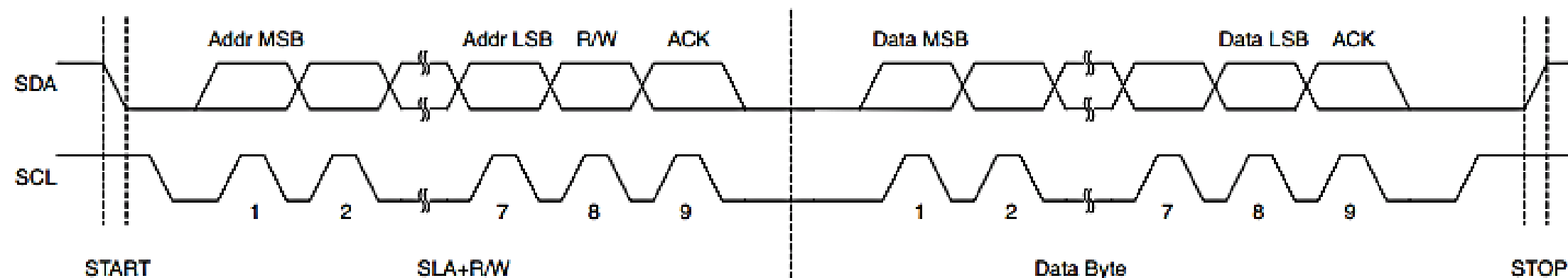


Data Packet Format

All data packets transmitted on the TWI bus are nine bits long, consisting of one data byte and an acknowledge bit. During a data transfer, the master generates the clock and the START and STOP conditions, while the receiver is responsible for acknowledging the reception. An Acknowledge (ACK) is signalled by the receiver pulling the SDA line low during the ninth SCL cycle. If the receiver leaves the SDA line high, a NACK is signalled.



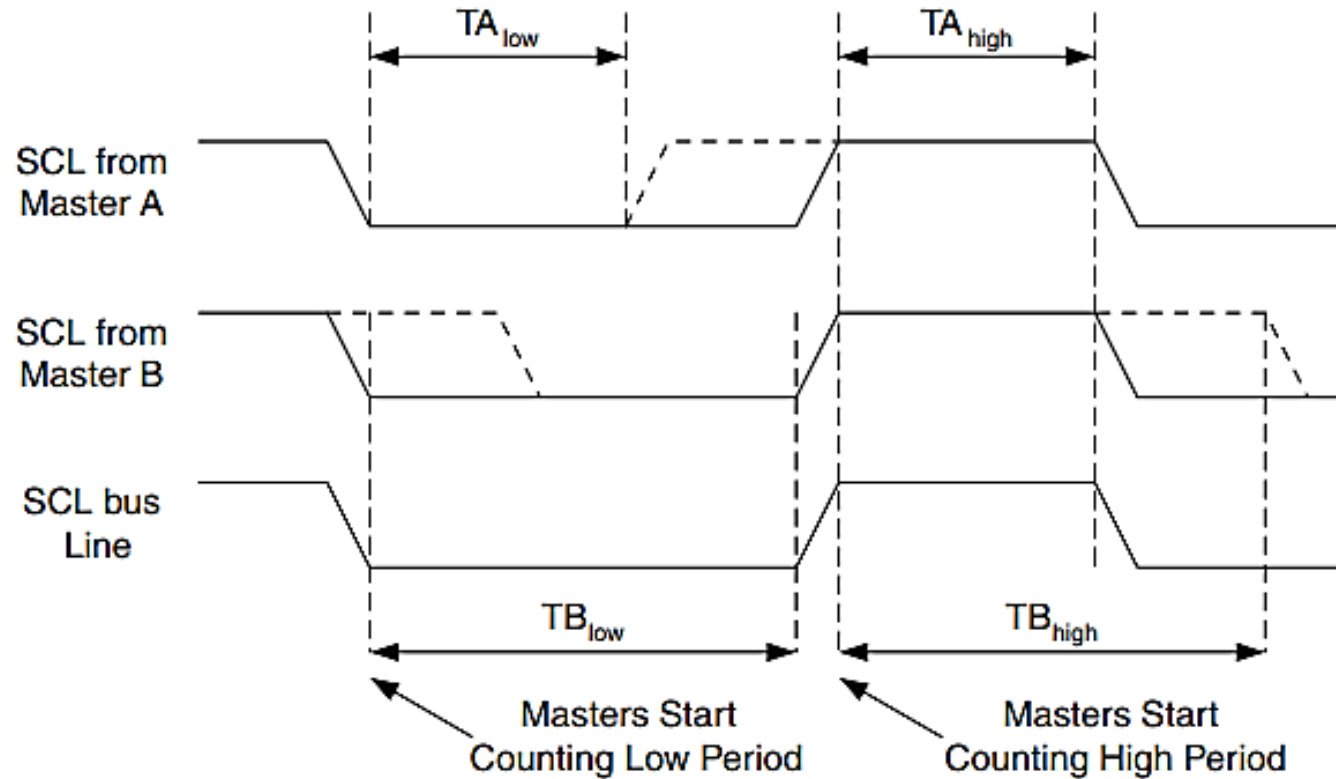
Combining Address and Data Packets into a Transmission



Typical Data Transmission

Multi-master Bus System

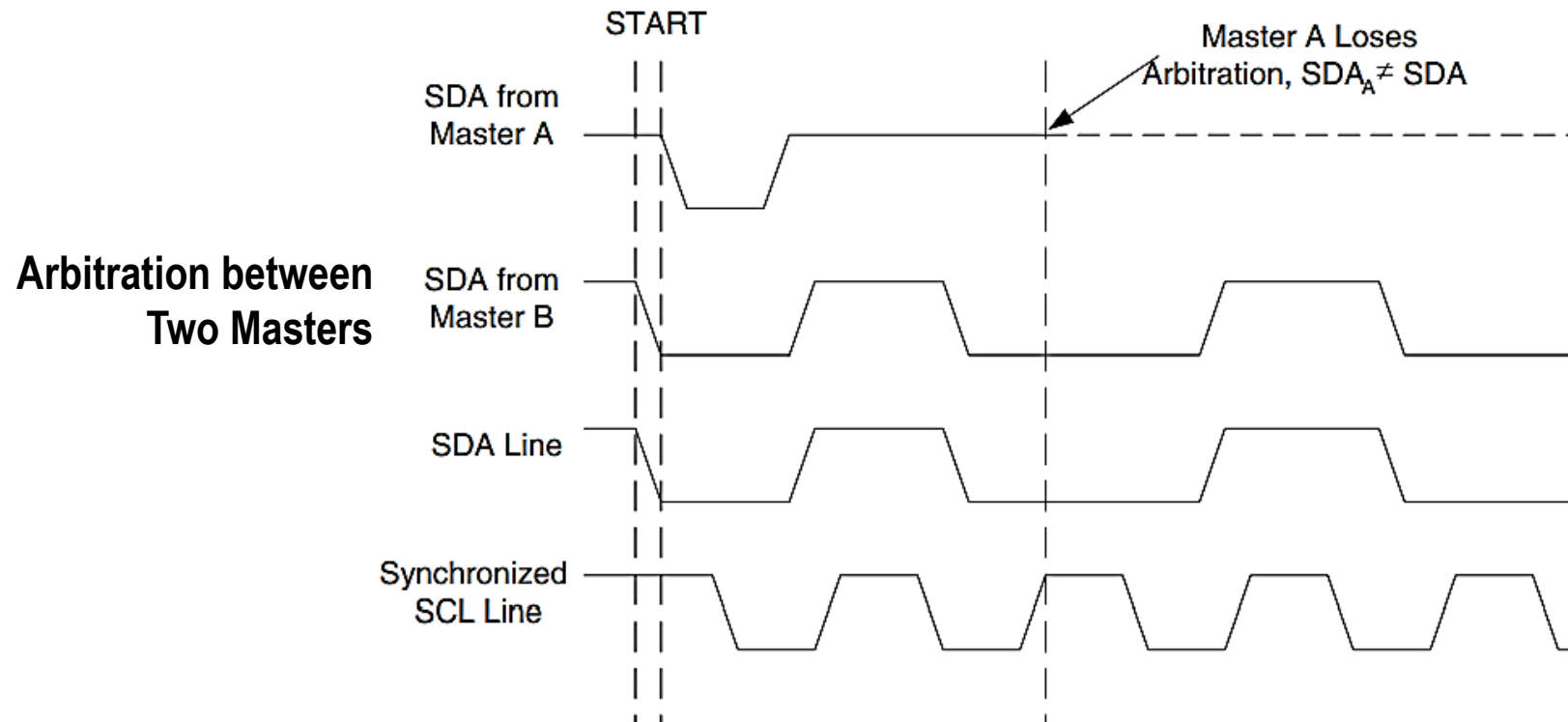
The TWI protocol allows bus systems with several masters. Special concerns have been taken in order to ensure that transmissions will proceed as normal, even if two or more masters initiate a transmission at the same time.



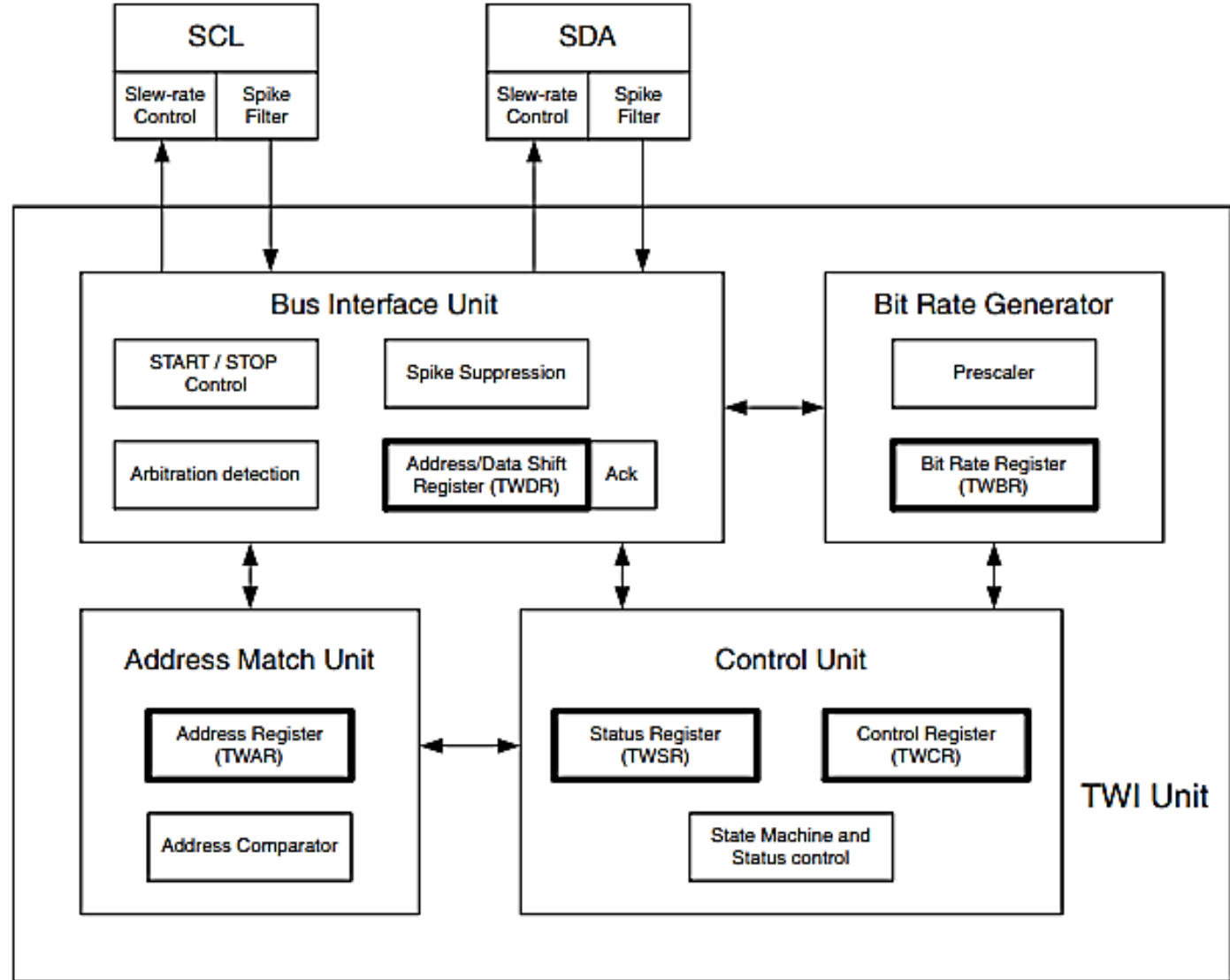
**SCL Synchronization
between Multiple
Masters**

Arbitration

- An algorithm must be implemented allowing only one of the masters to complete the transmission. All other masters should cease transmission when they discover that they have lost the selection process. This selection process is called arbitration.
- Arbitration is carried out by all masters continuously monitoring the SDA line after outputting data. If the value read from the SDA line does not match the value the master had output, it has lost the arbitration. Note that a master can only lose arbitration when it outputs a high SDA value while another master outputs a low value.



Overview of The TWI Module



Bit Rate Generator

This unit controls the period of SCL when operating in a Master mode. The SCL period is controlled by settings in the TWI Bit Rate Register (TWBR) and the Prescaler bits in the TWI Status Register (TWSR).

$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot 4^{\text{TWPS}}}$$

- TWBR = Value of the TWI Bit Rate Register
- TWPS = Value of the prescaler bits in the TWI Status Register

TWI Bit Rate Register - TWBR

Bit	7	6	5	4	3	2	1	0	
	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bits 7..0 – TWI Bit Rate Register**
 TWBR selects the division factor for the bit rate generator. The bit rate generator is a frequency divider which generates the SCL clock frequency in the Master modes.

TWI Control Register - TWCR

Bit	7	6	5	4	3	2	1	0	
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 7 – TWINT: TWI Interrupt Flag
- Bit 6 – TWEA: TWI Enable Acknowledge Bit
- Bit 5 – TWSTA: TWI START Condition Bit
- Bit 4 – TWSTO: TWI STOP Condition Bit
- Bit 3 – TWWC: TWI Write Collision Flag
- Bit 2 – TWEN: TWI Enable Bit
- Bit 1 – Res: Reserved Bit
- Bit 0 – TWIE: TWI Interrupt Enable

TWI Status Register - TWSR

Bit	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	1	1	1	1	1	0	0	0	

- Bits 7..3 – TWS: TWI Status
- Bit 2 – Res: Reserved Bit
- Bits 1..0 – TWPS: TWI Prescaler Bits

TWPS1	TWPS0	Prescaler Value
0	0	1
0	1	4
1	0	16
1	1	64

TWI Data Register - TWDR

Bit	7	6	5	4	3	2	1	0	
	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

- **Bits 7..0 – TWD: TWI Data Register**

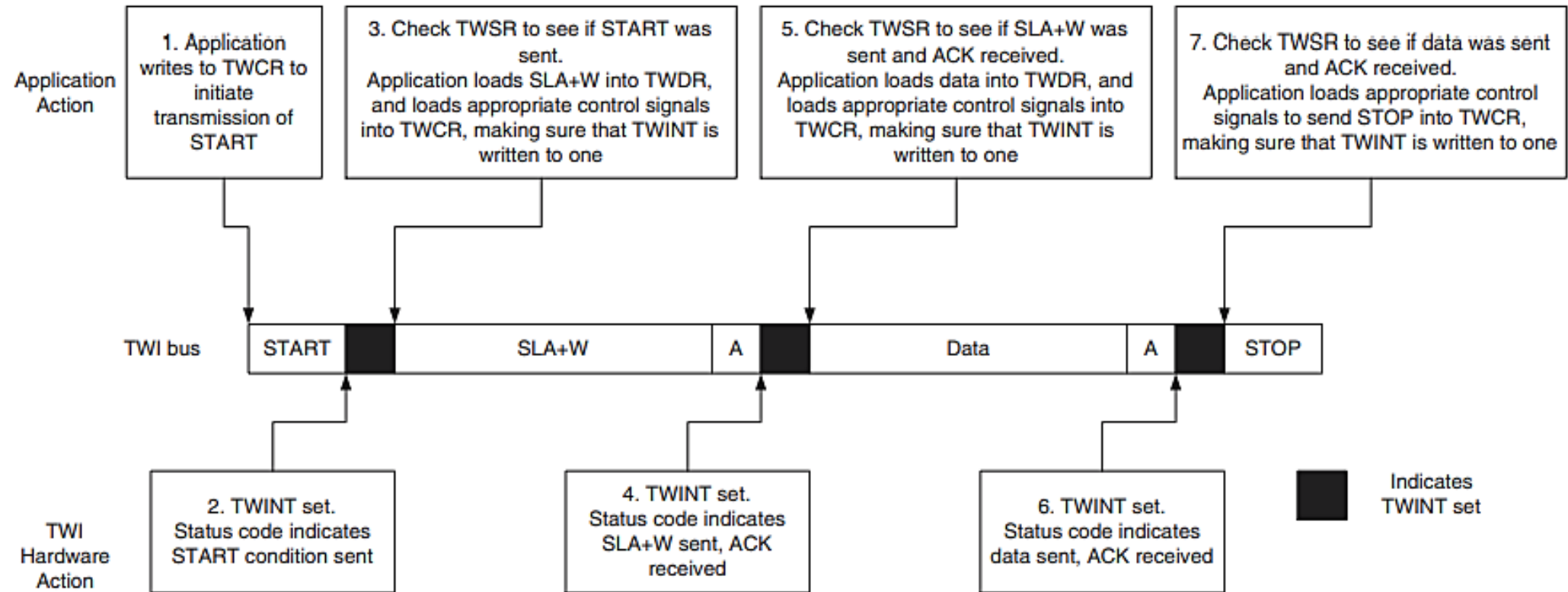
These eight bits contain the next data byte to be transmitted, or the latest data byte received on the Two-wire Serial Bus.

TWI (Slave) Address Register - TWAR

Bit	7	6	5	4	3	2	1	0	
	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	0	

- **Bits 7..1 – TWA: TWI (Slave) Address Register**
These seven bits constitute the slave address of the TWI unit.
- **Bit 0 – TWGCE: TWI General Call Recognition Enable Bit**
If set, this bit enables the recognition of a General Call given over the Two-wire Serial Bus.

Interfacing the Application to the TWI in a Typical Transmission



C example	Comments
<code>TWCR = (1<<TWINT) (1<<TWSTA) (1<<TWEN)</code>	Send START condition
<code>while (!(TWCR & (1<<TWINT))) ;</code>	Wait for TWINT flag set. This indicates that the START condition has been transmitted
<code>if ((TWSR & 0xF8) != START) ERROR();</code>	Check value of TWI Status Register. Mask prescaler bits. If status different from START go to ERROR
<code>TWDR = SLA_W; TWCR = (1<<TWINT) (1<<TWEN);</code>	Load SLA_W into TWDR Register. Clear TWINT bit in TWCR to start transmission of address
<code>while (!(TWCR & (1<<TWINT))) ;</code>	Wait for TWINT flag set. This indicates that the SLA+W has been transmitted, and ACK/NACK has been received.

<code>if ((TWSR & 0xF8) != MT_SLA_ACK) ERROR();</code>	Check value of TWI Status Register. Mask prescaler bits. If status different from MT_SLA_ACK go to ERROR
<code>TWDR = DATA; TWCR = (1<<TWINT) (1<<TWEN);</code>	Load DATA into TWDR Register. Clear TWINT bit in TWCR to start transmission of data
<code>while (!(TWCR & (1<<TWINT))) ;</code>	Wait for TWINT flag set. This indicates that the DATA has been transmitted, and ACK/NACK has been received.
<code>if ((TWSR & 0xF8) != MT_DATA_ACK) ERROR();</code>	Check value of TWI Status Register. Mask prescaler bits. If status different from MT_DATA_ACK go to ERROR
<code>TWCR = (1<<TWINT) (1<<TWEN) (1<<TWSTO);</code>	Transmit STOP condition

SEKIAN DAN TERIMA KASIH

PENGANTAR MIKROKONTROLER

Ahmad Zarkasi



MATERI BAHASAN

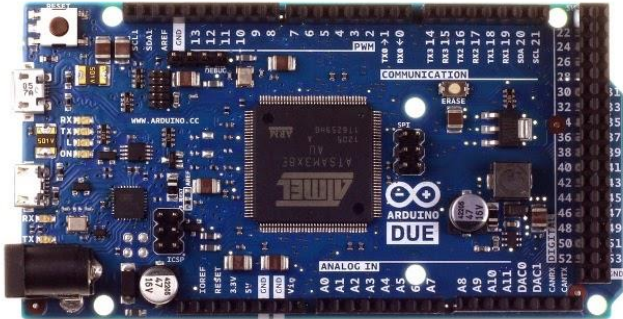
PEMROGRAMAN ARDUINO Bag. 1

Beberapa Jenis Modul Arduino (Review)

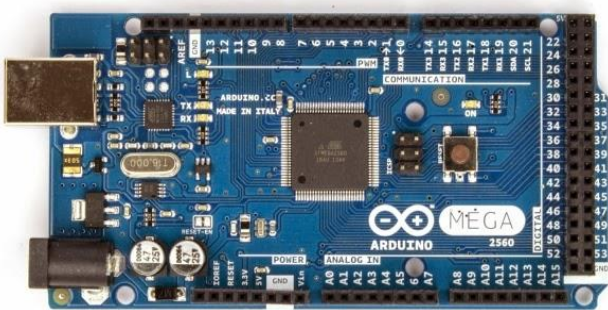
Arduino Uno (Topik Bahasan)



Arduino Due



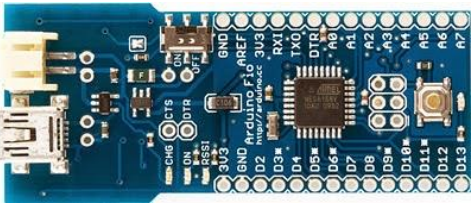
Arduino Mega



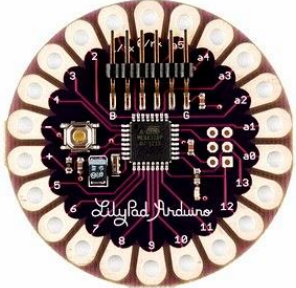
Arduino Leonardo



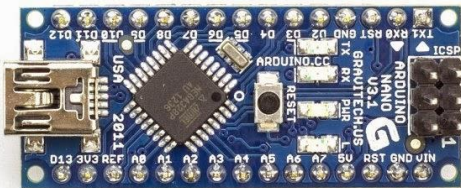
Arduino Fio



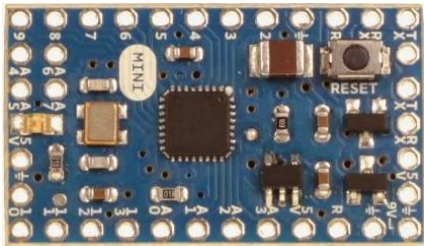
Arduino Lily



Arduino Nano



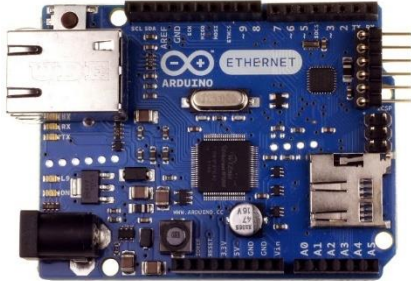
Arduino Mini



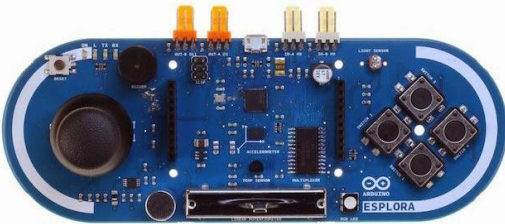
Arduino Micro



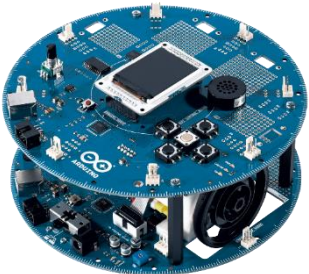
Arduino Ethernet



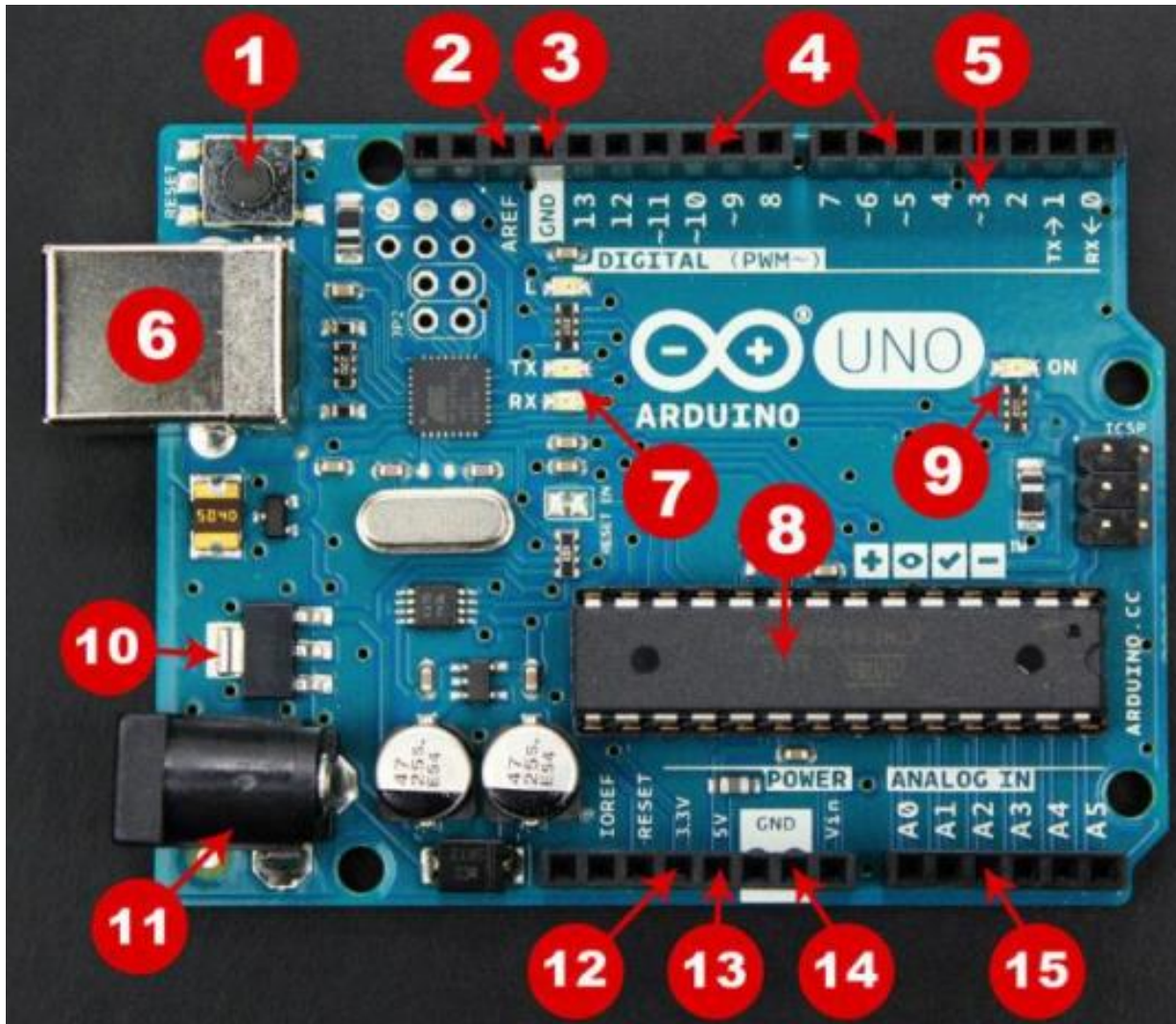
Arduino Esplora



Arduino Robot



Ringkasan Bagian-bagian Arduino Uno (Review)

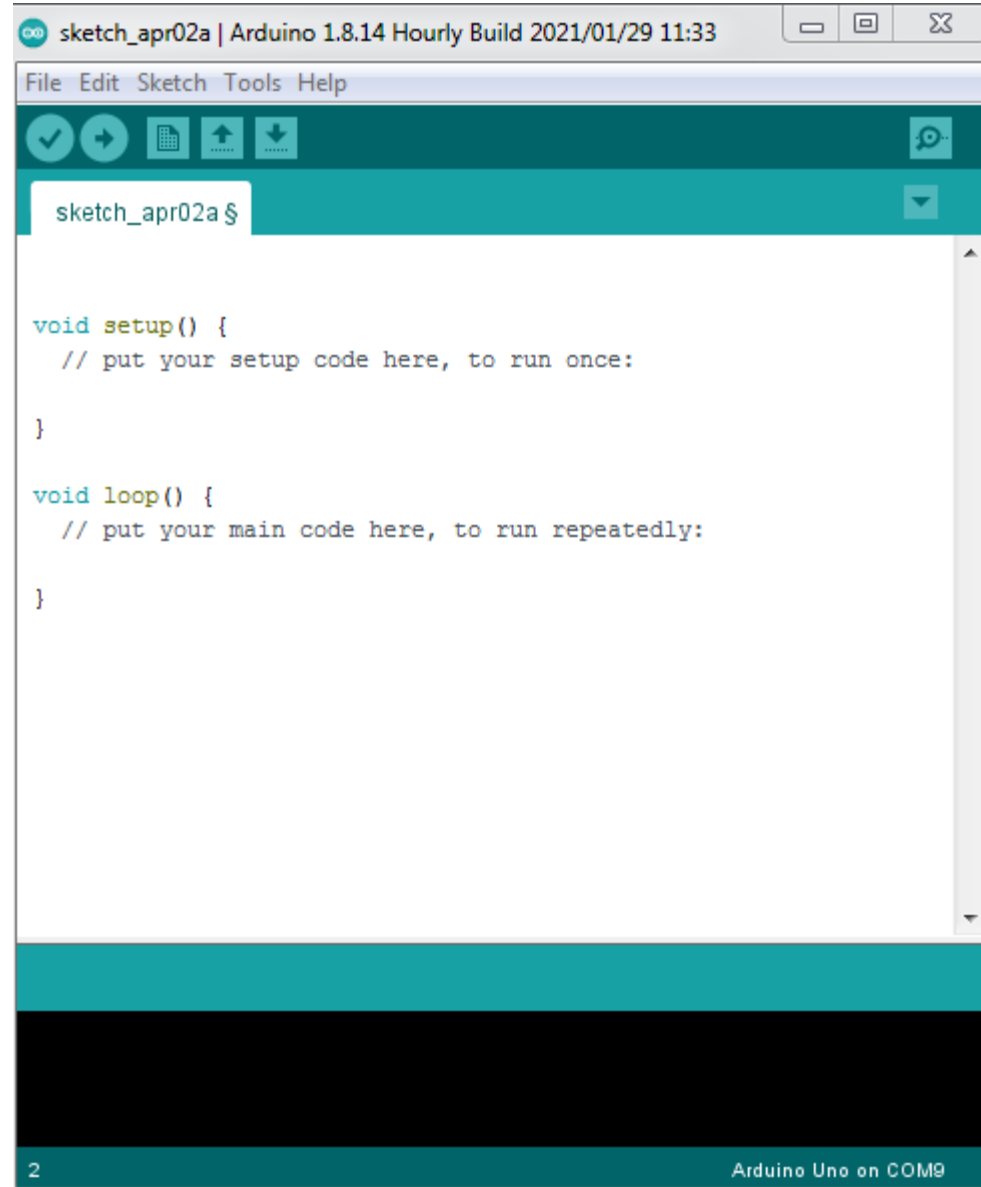


1. Reset Button
2. AREF
3. Ground Pin
4. Digital I/O
5. PWM
6. USB Connection
7. TX/RX
8. ATmega328P
9. Power LED Indicator
10. Voltage Regulator
11. DC Power Barrel Jack
12. 3.3V Pin
13. 5V Pin
14. Ground Pin
15. Analog Pin

Outline

1. Arduino IDE (Menu dan bagian-bagiannya)
2. Header, Void Setup, & Void Loop
3. Arduino Cheat Sheet
4. Tipe Data, Variabel, & Comment
5. digitalWrite (LED pada pin 13)
6. digitalWrite (4 LED)
7. For loop (4 LED)
8. Fungsi (4 LED)
9. Array 1D
10. digitalRead (Push Button)
11. Gerbang Logika AND & OR (2 Push Button, 3 LED)

1. Arduino IDE

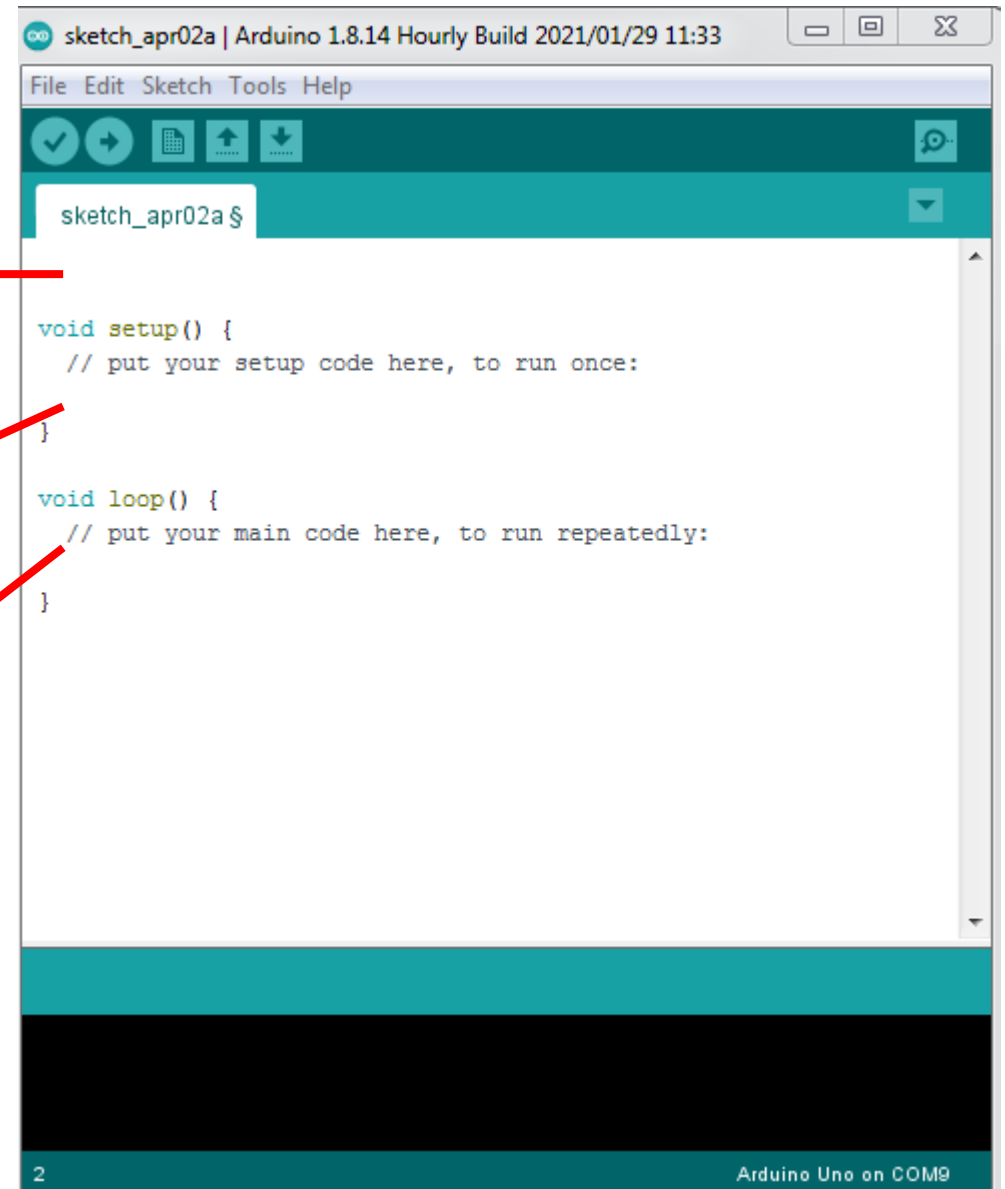


2. Header, Void Setup, & Void Loop

Header: Berisi deklarasi variabel, tipe data, dan lain-lain

Set up: Letak coding yang dieksekusi hanya sekali

Loop: Letak coding yang dieksekusi berulang-ulang



```
sketch_apr02a $  
  
void setup() {  
  // put your setup code here, to run once:  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
}
```

3. Arduino Cheat Sheet

Sketch Structures

- `setup()`
- `loop()`

Control Structures

- `if`
- `if...else`
- `for`
- `switch case`
- `while`
- `do...while`
- `break`
- `continue`
- `return`
- `goto`

Further Syntax

- `;` (semicolon)
- `{}` (curley braces)
- `//` (single line comment)
- `/**/` (multi-line comment)
- `#define`
- `#include`

Arithmetic Operators

- `=` (assignment operator)
- `+` (addition)
- `-` (subtraction)
- `/` (division)
- `%` (modulo)

Comparison Operators

- `==` (equal to)
- `!=` (not equal to)
- `<` (less than)
- `>` (greater than)
- `<=` (less than or equal to)
- `>=` (greater than equal to)

Boolean Operators

- `&&` (and)
- `||` (or)
- `!` (not)

Pointer Access Operators

- `*` deference operator
- `&` reference operator

Bitwise Operators

- `&` (bitwise and)
- `|` (bitwise or)
- `^` (bitwise xor)
- `~` (bitwise not)
- `<<` (bitshift left)
- `>>` (bitshift right)

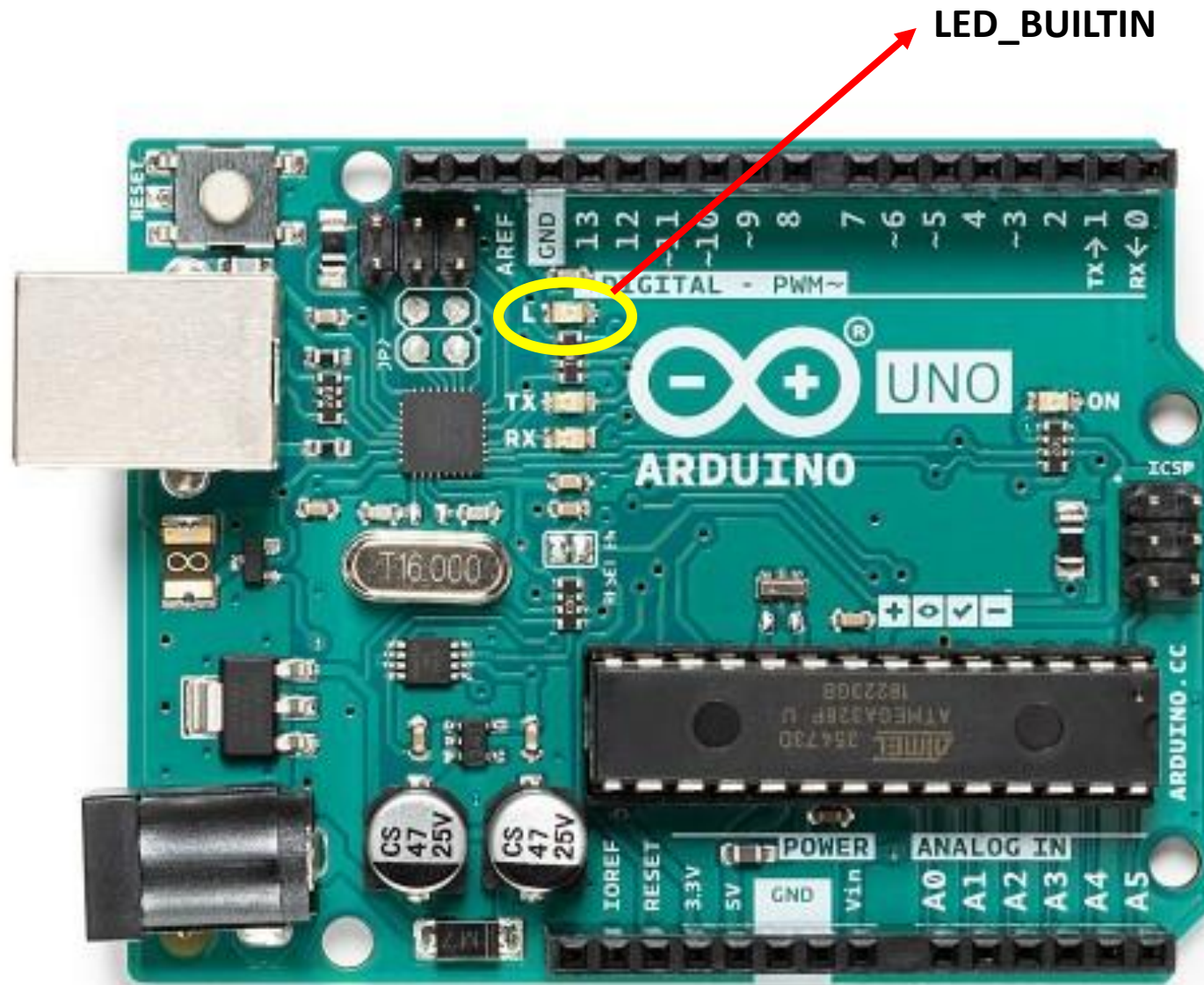
Compound Operators

- `++` (increment)
- `--` (decrement)
- `+=` (compound addition)
- `-=` (compound subtraction)
- `*=` (compound multiplication)
- `/=` (compound division)
- `%=` (compound modulo)
- `&=` (compound bitwise and)
- `!|=` (compound bitwise or)

4. Tipe Data, Variabel, dan Comment

Data Type	Size (Bytes)	Range of Values
void	0	null
bool/boolean	1	True/False
char	1	-128 to +127
unsigned char	1	0 to 255
byte	1	0 to 255
int	2	-32,768 to 32,767
unsigned int	2	0 to 65,535
word	2	0 to 65,535
long	4	-2,147,483,648 to 2,147,483,647
unsigned long	4	0 to 4,294,967,295
float	4	-3.4028235E+38 to 3.4028235E+38
double	4	-3.4028235E+38 to 3.4028235E+38
string	-	character array

5. digitalWrite (LED pada pin 13)



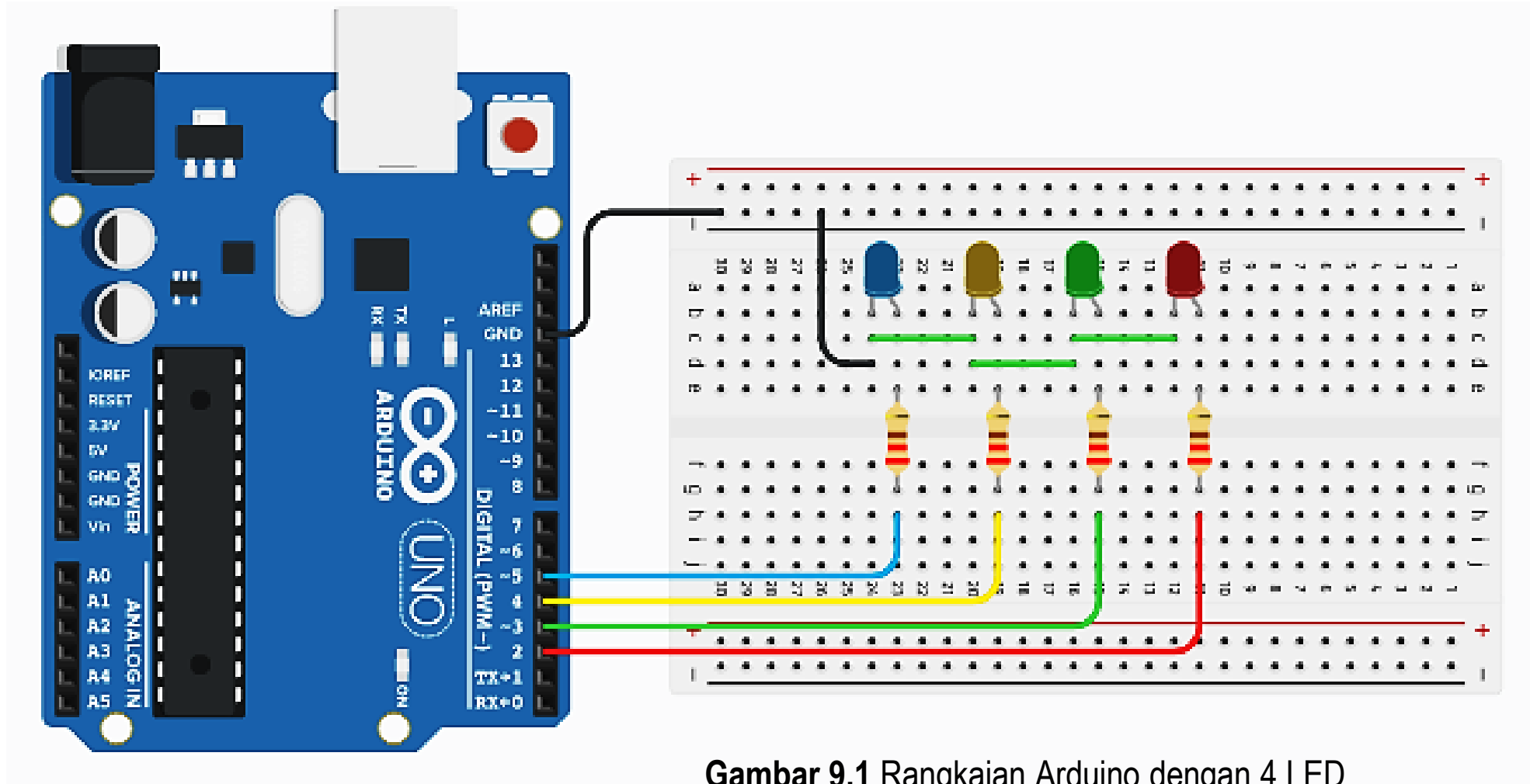
Code 1: Led Built in

```
int Led = 13; // LED_BUILTIN

void setup() {
  pinMode (Led, OUTPUT);
}

void loop() {
  digitalWrite(Led, HIGH);
  delay(300);
  digitalWrite(Led, LOW);
  delay(300);
}
```

6. digitalWrite (4 LED)



Gambar 9.1 Rangkaian Arduino dengan 4 LED

Code 2: 4 Led

```
const int Led1=2;
const int Led2=3;
const int Led3=4;
const int Led4=5;

int waktu = 500;

void setup() {
  pinMode(Led1, OUTPUT);
  pinMode(Led2, OUTPUT);
  pinMode(Led3, OUTPUT);
  pinMode(Led4, OUTPUT);
}

void loop() {
  digitalWrite(Led1, HIGH);
  delay (waktu);
  digitalWrite(Led1, LOW);
  delay (waktu);
```

```
  digitalWrite(Led2, HIGH);
  delay (waktu);
  digitalWrite(Led2, LOW);
  delay (waktu);
  digitalWrite(Led2, HIGH);
  delay (waktu);
  digitalWrite(Led2, LOW);
  delay (waktu);

  digitalWrite(Led3, HIGH);
  delay (waktu);
  digitalWrite(Led3, LOW);
  delay (waktu);
  digitalWrite(Led3, HIGH);
  delay (waktu);
  digitalWrite(Led3, LOW);
  delay (waktu);
  digitalWrite(Led3, HIGH);
  delay (waktu);
  digitalWrite(Led3, LOW);
  delay (waktu);
```

```
  digitalWrite(Led4, HIGH);
  delay (waktu);
  digitalWrite(Led4, LOW);
  delay (waktu);
  digitalWrite(Led4, HIGH);
  delay (waktu);
  digitalWrite(Led4, LOW);
  delay (waktu);
  digitalWrite(Led4, HIGH);
  delay (waktu);
  digitalWrite(Led4, LOW);
  delay (waktu);
}
```

7. For Loop (4 LED)

Code 3: For Loop

Script sebelumnya (Code 2) terlihat sangat panjang dan tentu tidak efisien. Untuk mempersingkatnya kita dapat menggunakan for loop.

Syntax

```
for (initialization; condition; increment) {  
    // statement(s);  
}
```

```
const int Led1=2;  
const int Led2=3;  
const int Led3=4;  
const int Led4=5;  
int waktu=300;  
int i;  
  
void setup() {  
    pinMode(Led1, OUTPUT);  
    pinMode(Led2, OUTPUT);  
    pinMode(Led3, OUTPUT);  
    pinMode(Led4, OUTPUT);  
  
}
```

```
void loop() {  
  
    for(i=1; i<=1; i++){  
        digitalWrite(Led1, HIGH);  
        delay(waktu);  
        digitalWrite(Led1, LOW);  
        delay(waktu);  
    }  
  
    for(i=1; i<=2; i++){  
        digitalWrite(Led2, HIGH);  
        delay(waktu);  
        digitalWrite(Led2, LOW);  
        delay(waktu);  
    }  
  
}
```

```
for(i=1; i<=3; i++){  
    digitalWrite(Led3, HIGH);  
    delay(waktu);  
    digitalWrite(Led3, LOW);  
    delay(waktu);  
}  
  
for(i=1; i<=4; i++){  
    digitalWrite(Led4, HIGH);  
    delay(waktu);  
    digitalWrite(Led4, LOW);  
    delay(waktu);  
}  
}
```

8. Fungsi (4 LED)

Code 4: Contoh Fungsi

Jika Code 3 masih dirasa belum ringkas, kita dapat membuat fungsi tersendiri, yang nantinya akan dipanggil di void loop

```
const int Led1=2;
const int Led2=3;
const int Led3=4;
const int Led4=5;
int i;

void setup() {
  pinMode(Led1, OUTPUT);
  pinMode(Led2, OUTPUT);
  pinMode(Led3, OUTPUT);
  pinMode(Led4, OUTPUT);
}

void loop() {

  kedip (1, 1, Led1, 500);
  kedip (1, 2, Led2, 400);
  kedip (1, 3, Led3, 300);
  kedip (1, 4, Led4, 200);

}
```

```
void kedip (int nilaiAwal, int
nilaiAkhir, int Led, int waktu){

  for (i=nilaiAwal; i<=nilaiAkhir; i++){
    digitalWrite(Led, HIGH);
    delay (waktu);
    digitalWrite(Led, LOW);
    delay (waktu);
  }

}
```

Jika diperhatikan, script di samping menjadi lebih ringkas dan fleksibel karena pengaturan berapa kali kedip pada led hanya perlu mengganti parameter yang ada di fungsi **kedip** .

Selain menggunakan fungsi, script ini juga dapat dipersingkat dengan menggunakan array.

9. Array 1D

Code 5: Contoh Array 1D

Instruksi:

Buatlah script menggunakan Arduino IDE untuk mensimulasikan 4 led yang nyala secara bergantian menggunakan array 1D.

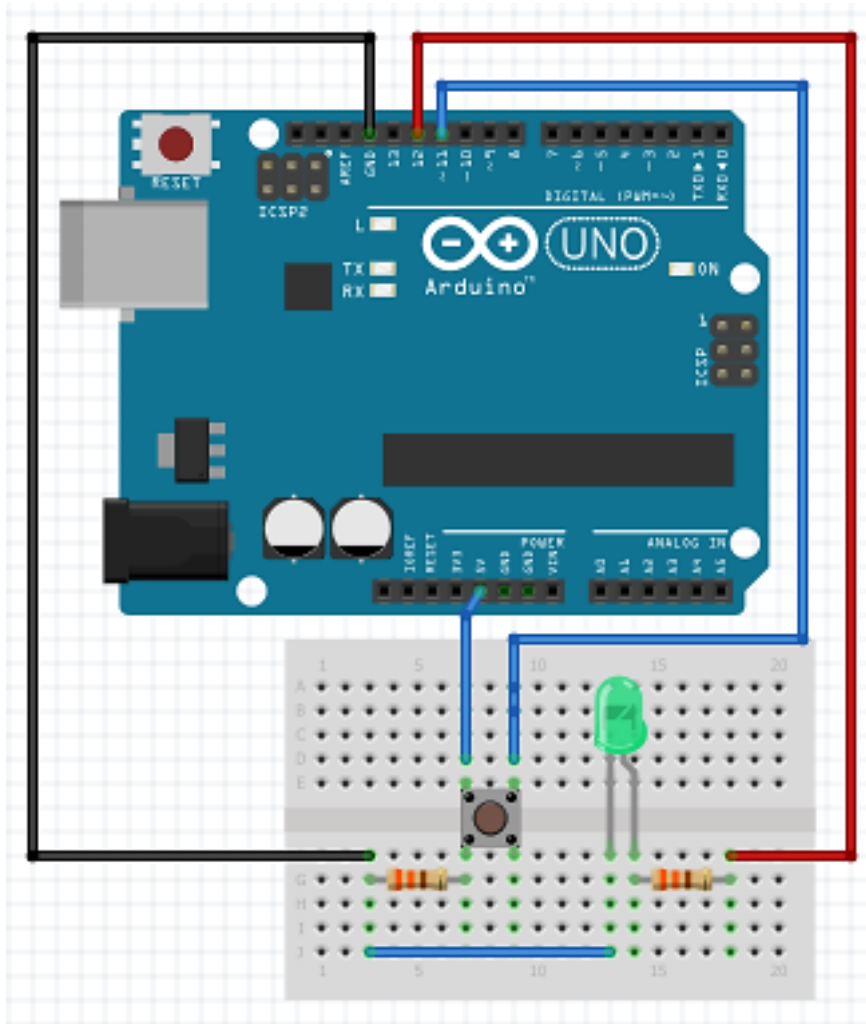
Note: Gunakan skema rangkaian 9.1

```
int led[4]={5, 4, 3, 2}; //indeks dimulai dari 0, berarti 0 s.d 4

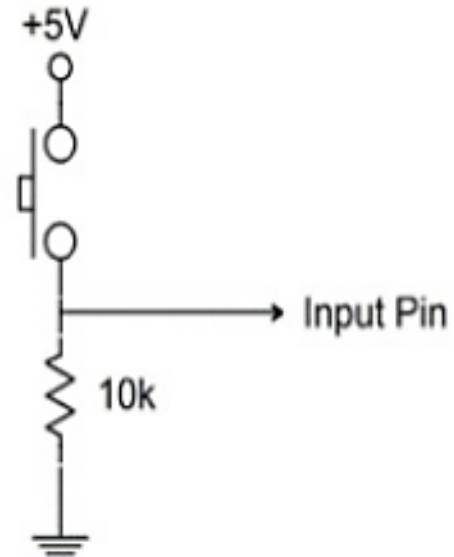
void setup(){
  for (int i=0; i<=3; i++){
    pinMode(led[i], OUTPUT);
  }
}

void loop(){
  for (int i=0; i<=3; i++){
    digitalWrite(led[i], HIGH);
    delay(1000);
    digitalWrite(led[i], LOW);
    delay(1000);
  }
}
```

9. digitalRead (1 Push Button)

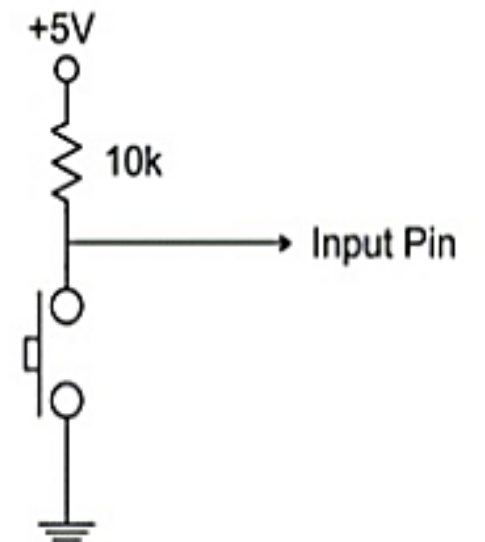


Resistor Pull Down



Default (LOW)
Merupakan rangkaian yang digunakan

Resistor Pull Up



Default (HIGH)

Code 6: Push Button

```
const int Led = 12;
const int Pb = 11;

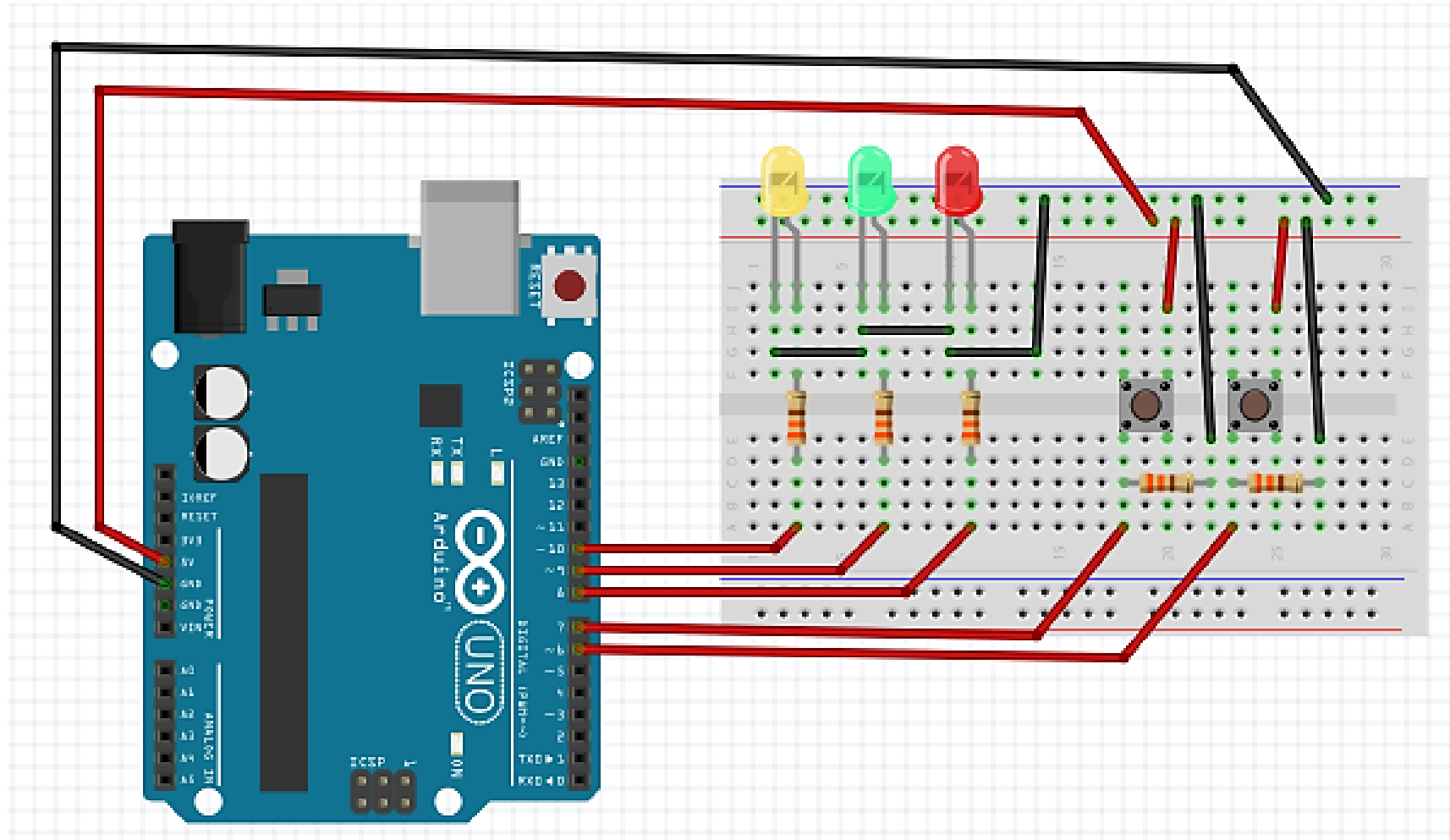
void setup() {
  pinMode (Led, OUTPUT);
  pinMode (Pb, INPUT);
  Serial.begin(9600);
}

void loop() {
  int nilaiPb = digitalRead (Pb);

  if(nilaiPb==HIGH){
    digitalWrite (Led, HIGH);
  }else{
    digitalWrite (Led, LOW);
  }

  Serial.println(nilaiPb);
}
```

10. Gerbang Logika AND & OR (2 Push Button, 3 LED)



Code 7: Logika AND

```
const int ledM=12;
const int ledH=11;
const int ledK=10;
const int pb1=7; //pb Merah
const int pb2=6; //pb Hijau
int nilaipb1;
int nilaipb2;

void setup() {
  Serial.begin(9600);
  pinMode(ledM, OUTPUT);
  pinMode(ledH, OUTPUT);
  pinMode(ledK, OUTPUT);
  pinMode(pb1, INPUT);
  pinMode(pb2, INPUT);
}

void loop() {
  nilaipb1=digitalRead(pb1);
  nilaipb2=digitalRead(pb2);
```

```
  if(nilaipb1==0 && nilaipb2==0){
    digitalWrite(ledM, HIGH);
    digitalWrite(ledH, HIGH);
    digitalWrite(ledK, HIGH);
  }

  if(nilaipb1==1 && nilaipb2==0){
    digitalWrite(ledM, LOW);
    digitalWrite(ledH, HIGH);
    digitalWrite(ledK, LOW);
  }

  if(nilaipb1==0 && nilaipb2==1){
    digitalWrite(ledM, HIGH);
    digitalWrite(ledH, LOW);
    digitalWrite(ledK, LOW);
  }

  if(nilaipb1==0 && nilaipb2==0){
    digitalWrite(ledM, HIGH);
    digitalWrite(ledH, HIGH);
    digitalWrite(ledK, LOW);
  }
```

```
  if(nilaipb1==1 && nilaipb2==0){
    digitalWrite (ledM, HIGH);
  }else{
    digitalWrite (ledM, LOW);
  }

  if(nilaipb1==0 && nilaipb2==1){
    digitalWrite (ledH, HIGH);
  }else{
    digitalWrite (ledH, LOW);
  }

  if(nilaipb1==1 && nilaipb2==1){
    digitalWrite (ledK, HIGH);
  }else{
    digitalWrite (ledK, LOW);
  }

  Serial.print(nilaipb1);
  Serial.print(" | ");
  Serial.println(nilaipb2);
}
```

Code 8: Logika OR

<pre>const int ledM=12; const int ledH=11; const int ledK=10; const int pb1=7; //pb Merah const int pb2=6; //pb Hijau int nilaipb1; int nilaipb2; void setup() { Serial.begin(9600); pinMode(ledM, OUTPUT); pinMode(ledH, OUTPUT); pinMode(ledK, OUTPUT); pinMode(pb1, INPUT); pinMode(pb2, INPUT); } void loop() { nilaipb1=digitalRead(pb1); nilaipb2=digitalRead(pb2);</pre>	<pre>if(nilaipb1==1 nilaipb2==1){ digitalWrite(ledK, HIGH); }else{ digitalWrite(ledK, LOW); } Serial.print(nilaipb1); Serial.print(" "); Serial.println(nilaipb2); }</pre>
---	---

PENGANTAR MIKROKONTROLER

Ahmad Zarkasi



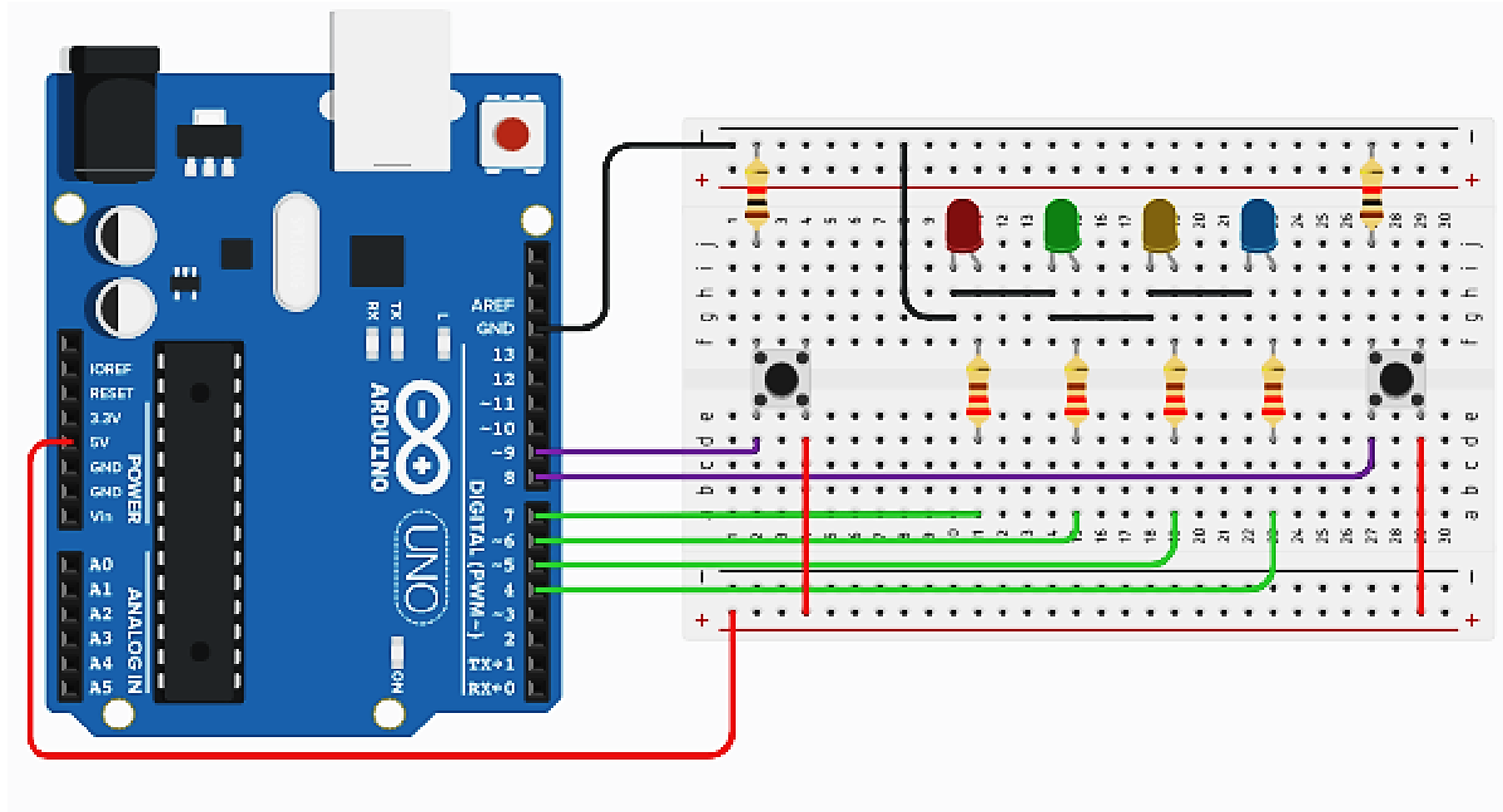
MATERI BAHASAN

PEMROGRAMAN ARDUINO Bag. 2

Outline

1. Switch Case
2. analogWrite (PWM)
3. Libraries
4. Sensor & Module
5. Coding Using Library
6. Coding Without Library

1. Switch Case



Code 9: Switch Case

```
const int led[4] = {7,6,5,4};
const int pb[2] = {8,9};
int j;

void setup(){

    for (int i=0; i<=3; i++){
        pinMode(led[i], OUTPUT);
    }

    for (int i=0; i<=1; i++){
        pinMode(pb[i], INPUT);
    }

    Serial.begin(9600);
}

void loop(){

    int nilaiPb1 =
digitalRead(pb[0]);
    int nilaiPb2 =
digitalRead(pb[1]);
```

```
    if(nilaiPb1==HIGH){
        j++;
        delay(20);
    }
    if(nilaiPb2==HIGH){
        j--;
        delay(20);
    }

    switch (j){

        case 1:
            digitalWrite(led[0], HIGH);
            digitalWrite(led[1], LOW);
            digitalWrite(led[2], LOW);
            digitalWrite(led[3], LOW);
            break;

        case 2:
            digitalWrite(led[0], LOW);
            digitalWrite(led[1], HIGH);
            digitalWrite(led[2], LOW);
            digitalWrite(led[3], LOW);
            break;
```

```
        case 3:
            digitalWrite(led[0], LOW);
            digitalWrite(led[1], LOW);
            digitalWrite(led[2], HIGH);
            digitalWrite(led[3], LOW);
            break;

        case 4:
            digitalWrite(led[0], LOW);
            digitalWrite(led[1], LOW);
            digitalWrite(led[2], LOW);
            digitalWrite(led[3], HIGH);
            break;

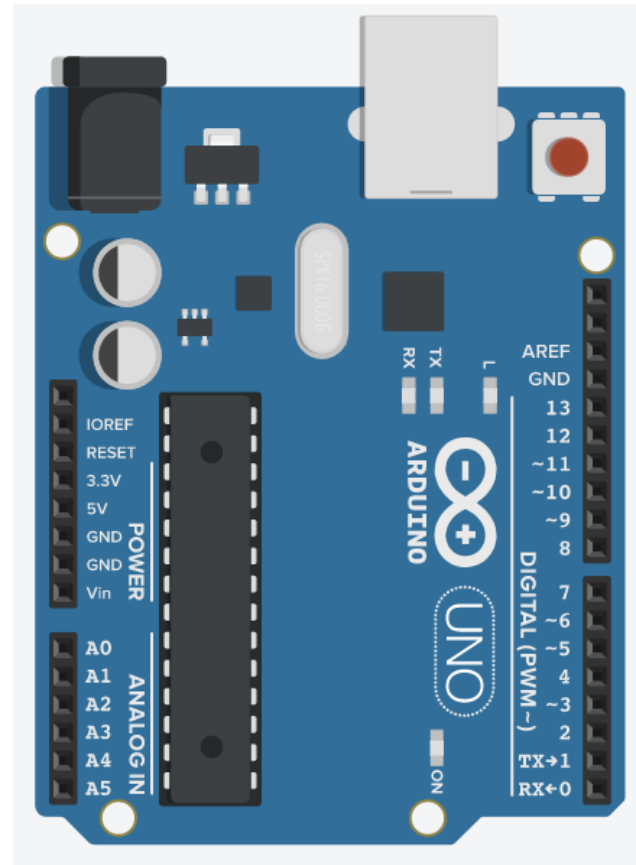
        default:
            digitalWrite(led[0], LOW);
            digitalWrite(led[1], LOW);
            digitalWrite(led[2], LOW);
            digitalWrite(led[3], LOW);
            break;
    }

    Serial.println(j);
}
```

2. PWM

Pulse width modulation (PWM), or **pulse-duration modulation (PDM)**, is a method of reducing the average power delivered by an electrical signal, by effectively chopping it up into discrete parts.

Amplitude & Frequency : Constant



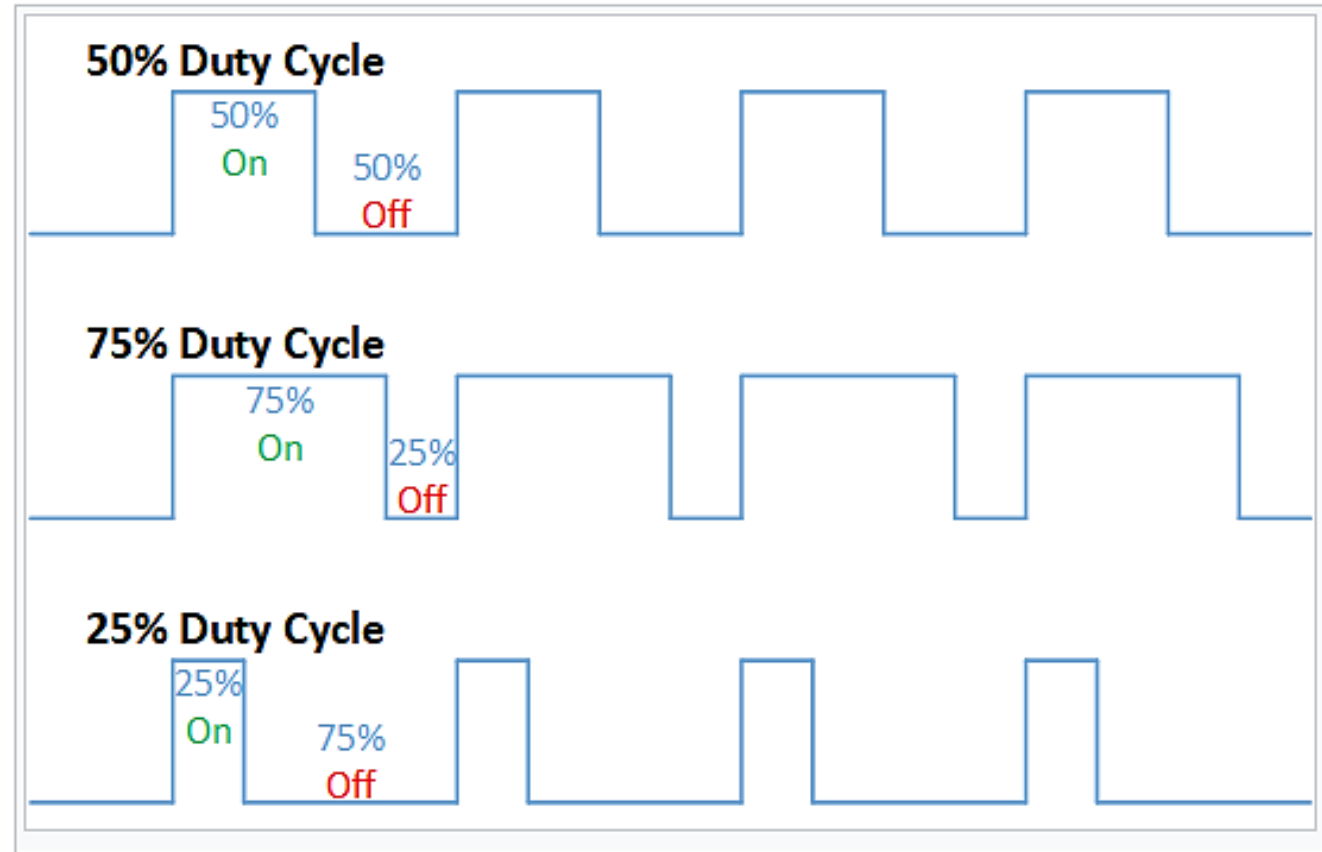
In Arduino UNO, PWM available on:

- Pin 3
- Pin 5
- Pin 6
- Pin 9
- Pin 10
- Pin 11

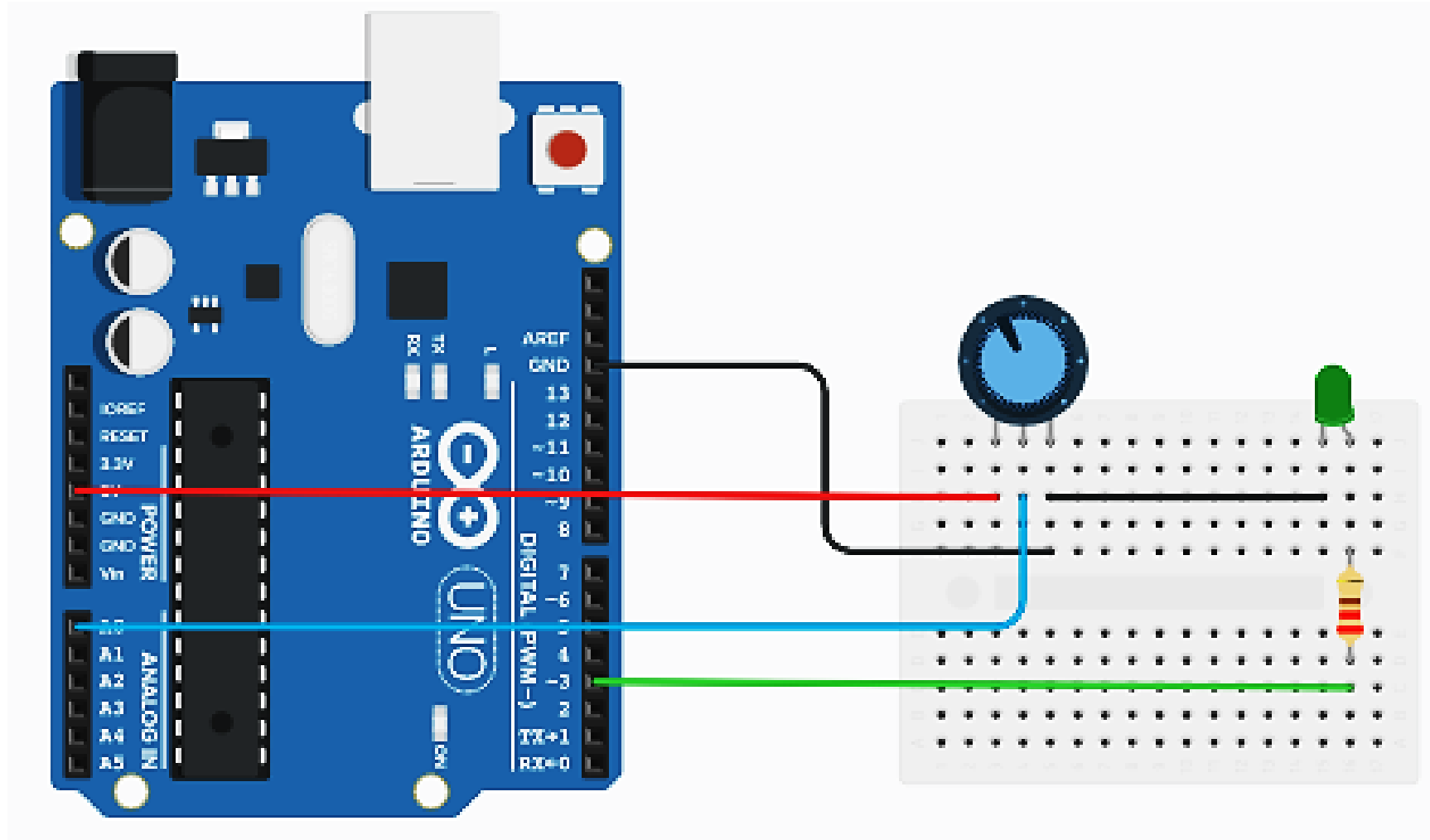
Duty Cycle

$$\text{DUTY} = \frac{T_{\text{HIGH}}}{T_{\text{TOTAL}}} \times 100\%$$

$$V_{\text{OUT}} = \text{DUTY} \times \text{Amplitude}$$



Contoh



Code 10: PWM

3. Libraries

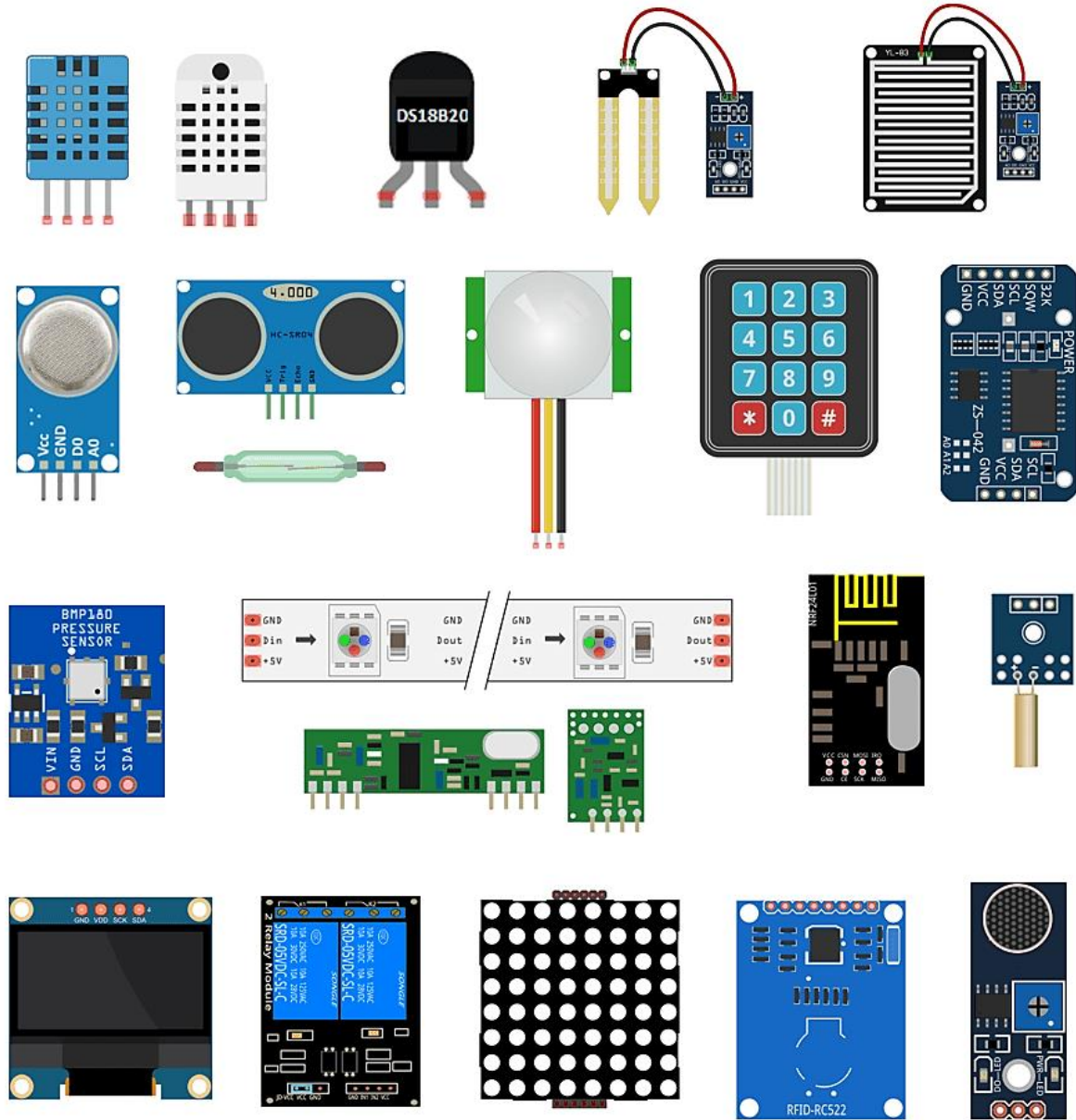
The Arduino environment can be extended through the use of libraries, just like most programming platforms. Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from Sketch > Import Library.

A number of libraries come installed with the IDE, but you can also download or create your own. See these instructions for details on installing libraries. There's also a tutorial on writing your own libraries. See the API Style Guide for information on making a good Arduino-style API for your library.

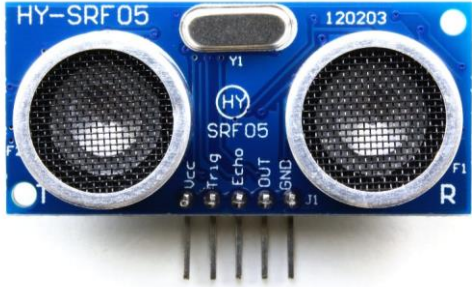
For more details:

<https://www.arduino.cc/en/reference/libraries>

4. Sensor & Module



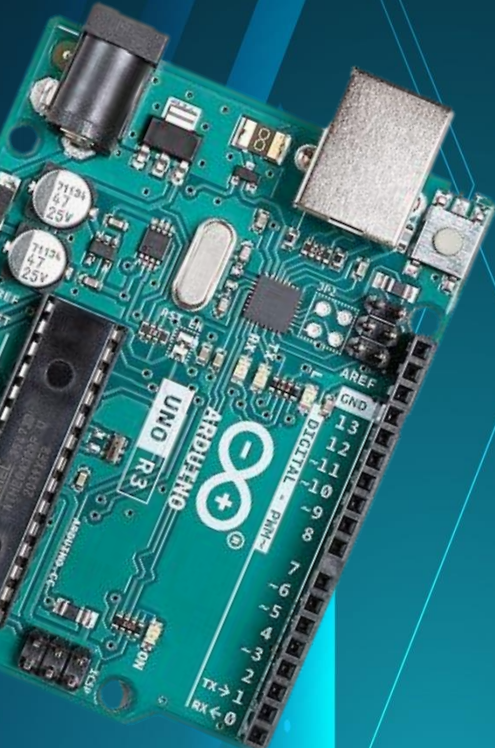
5. Coding Using Library



Ultrasonic Sensor

AHMAD ZARKASI

SIMULASI MIKROKONTROLER ARDUINO BERBASIS TINKERCAD



KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT atas rahmat dan karuniaNya, sehingga buku *Simulasi Mikrokontroler Arduino Berbasis Tinkercad* ini dapat terselesaikan. Penyusunan buku ini dilatarbelakangi oleh perlunya memperkenalkan media alternatif bagi kalangan yang ingin mempelajari mikrokontroler Arduino bahkan dari *basic* sekalipun. Media tersebut berupa *platform* simulasi yang bernama Tinkercad. Tinkercad merupakan *platform* simulasi *online* berbasis website. Oleh karena itu, pengguna tidak perlu melakukan instalasi di komputer. Karena menggunakan sistem *online*, maka Tinkercad juga dapat diakses melalui *smartphone*, tablet, dan lain sebagainya, sehingga menjadi lebih fleksibel.

Selain berisi petunjuk simulasi mikrokontroler, buku ini dilengkapi dengan beberapa materi penunjang yang penting khususnya bagi para pemula seperti ulasan tentang Mikroprosesor, Mikrokontroler dan Arduino; instalasi *software* Arduino IDE dan *platform* Tinkercad; serta beberapa *syntax* dasar pada pemrograman Arduino. Percobaan-percobaan yang dimuat pada buku ini disusun berdasarkan komponen-komponen yang tersedia pada *platform* Tinkercad dan tentu saja mudah didapatkan di pasaran.

Rampungnya penulisan buku ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu penulis menyampaikan terima kasih, khususnya kepada Bapak Dr. Djayus, M.T (Ketua Jurusan Fisika FMIPA UNMUL), Ibu Dr. Rahmawati Munir, M.Si (Koordinator Program Studi Fisika

FMIPA UNMUL), Dr. Syahrir, M.Si (Kepala Lab. Instrumentasi Fisika FMIPA UNMUL) yang turut memberikan saran dan masukan yang berharga.

Terakhir namun yang terpenting, penulis menyampaikan ucapan terima kasih yang mendalam kepada orang tua penulis (Ibu Zeniyah dan Bapak Haji Marjan), serta kepada kakak dan adik penulis (Hilmi, Zayyin, dan Julia), atas dukungan dan do'a sehingga penulis tetap konsisten dalam mengerjakan buku ini hingga selesai. Akhirnya, semoga buku ini bermanfaat bagi banyak orang.

Samarinda, Oktober 2021

Penulis

DAFTAR ISI

KATA PENGANTAR i

DAFTAR ISI iii

DAFTAR GAMBAR v

DAFTAR TABEL viii

BAB I. MIKROKONTROLER PLATFORM ARDUINO 1

- 1.1. MIKROKONTROLER 1
- 1.2. ARDUINO 3
- 1.3. ARDUINO IDE 10
- 1.4. TAHAPAN PERCOBAAN MENGGUNAKAN ARDUINO 19

BAB II. PLATFORM TINKERCAD 24

- 2.1. PENGENALAN TINKERCAD 25
- 2.2. REGISTRASI, PEMBUATAN CLASS, DAN JOIN CLASS DI TINKERCAD 30
- 2.3. PERCOBAAN ARDUINO DI TINKERCAD 41

BAB III. SYNTAX DASAR ARDUINO 47

- 3.1. STRUCTURE 47
- 3.2. VARIABLES (VARIABEL) 52
- 3.3. DATA TYPES (TIPE DATA) 54
- 3.4. OPERATOR 57
- 3.5. CONSTANTS (KONSTANTA) 64

3.6.	FLOW CONTROL	66
3.7.	DIGITAL I/O	70
3.8.	ANALOG I/O	72
3.9.	TIME	73
3.10.	MATH	74
3.11.	SERIAL	74
BAB IV. PERCOBAAN MIKROKONTROLER ARDUINO		77
4.1.	PERCOBAAN 1 – LED	78
4.2.	PERCOBAAN 2 – PUSH BUTTON	82
4.3.	PERCOBAAN 3 – ADC dan PWM	85
4.4.	PERCOBAAN 4 – LCD	88
4.5.	PERCOBAAN 5 – KEYPAD	91
4.6.	PERCOBAAN 6 – SENSOR TEMPERATUR	94
4.7.	PERCOBAAN 7 – SENSOR ULTRASONIK	97
4.8.	PERCOBAAN 8 – CONTOH PROJECT	100
REFERENSI		105

DAFTAR GAMBAR

Gambar 1.1. Arsitektur AVR ATmega16	3
Gambar 1.2. Beberapa tipe <i>board</i> Arduino	7
Gambar 1.3. Arduino dengan <i>chip</i> (a) tipe DIP dan (b) tipe SMD	8
Gambar 1.4. Menu pada arduino.cc	11
Gambar 1.5. Opsi <i>download</i>	11
Gambar 1.6. Memulai proses <i>download</i>	12
Gambar 1.7. <i>License Agreement</i>	12
Gambar 1.8. <i>Installation Options</i>	13
Gambar 1.9. <i>Installation Folder</i>	13
Gambar 1.10. Proses ekstraksi dan instalasi	14
Gambar 1.11. Proses instalasi selesai	14
Gambar 1.12. Jendela <i>software</i> Arduino IDE	15
Gambar 1.13. Contoh rangkaian LED sederhana	20
Gambar 1.14. Memilih <i>board</i> Arduino	21
Gambar 1.15. Memilih <i>Port</i>	21
Gambar 1.16. Proses <i>compile</i> program	23
Gambar 1.17. Proses <i>upload</i> program	24
Gambar 2.1. Tampilan awal web Tinkercad	25
Gambar 2.2. Contoh desain 3D menggunakan 3D Designs (a) dan Codeblocks (b)	27
Gambar 2.3. Contoh rangkaian sederhana di menu Circuit	28
Gambar 2.4. Menu pada tampilan awal Tinkercad	31
Gambar 2.5. Create a personal account di Tinkercad	31
Gambar 2.6. Pilihan mode untuk pendaftaran Tinkercad	31
Gambar 2.7. Memasukkan alamat email	32

Gambar 2.8. Memasukkan <i>password</i>	32
Gambar 2.9. Konfirmasi promosi dari Autodesk	32
Gambar 2.10. Tampilan awal pada akun Tinkercad	33
Gambar 2.11. Klik profil	34
Gambar 2.12. Select role	34
Gambar 2.13. Become an educator	35
Gambar 2.14. Keterangan status berubah menjadi teacher	35
Gambar 2.15. Fitur Classes	35
Gambar 2.16. Create new classes	36
Gambar 2.17. Contoh class	36
Gambar 2.18. Contoh class yang sudah dibuat	37
Gambar 2.19. Contoh kode unik (class code)	37
Gambar 2.20. Add students	38
Gambar 2.21. Contoh pengisian Name dan Nickname	38
Gambar 2.22. Join class di Tinkercad	39
Gambar 2.23. Memasukkan class code	39
Gambar 2.24. Muncul nama class	40
Gambar 2.25. Masukkan nick name	40
Gambar 2.26. Tampilan Tinkercad melalui join class	41
Gambar 2.27. Create new Circuit	42
Gambar 2.28. Components	42
Gambar 2.29. Komponen yang diperlukan untuk simulasi	43
Gambar 2.30. Mengubah nilai resistor dan warna LED	43
Gambar 2.31. Rangkaian simulasi	44
Gambar 2.32. Memunculkan text editor	44
Gambar 2.33. Start Simulation	46
Gambar 4.1. Rangkaian Percobaan 1	78
Gambar 4.2. Rangkaian Percobaan 2	82
Gambar 4.3. Rangkaian Percobaan 3	85

Gambar 4.4. Rangkaian Percobaan 4	88
Gambar 4.5. Rangkaian Percobaan 5	91
Gambar 4.6. Rangkaian Percobaan 6	94
Gambar 4.7. Rangkaian Percobaan 7	97
Gambar 4.8. Rangkaian Percobaan 8	101

DAFTAR TABEL

Tabel 1.1. Perbedaan Mikrokontroler dengan Mikroprosesor 1

Tabel 3 1. Jenis Tipe Data 57

Tabel 3.2. Operator Aritmatika 58

Tabel 3.3. Operator Pembanding 59

Tabel 3.4. Operator Logika 61

Tabel 3.5. Operator Majemuk 62

BAB I

MIKROKONTROLER PLATFORM ARDUINO

1.1 MIKROKONTROLER

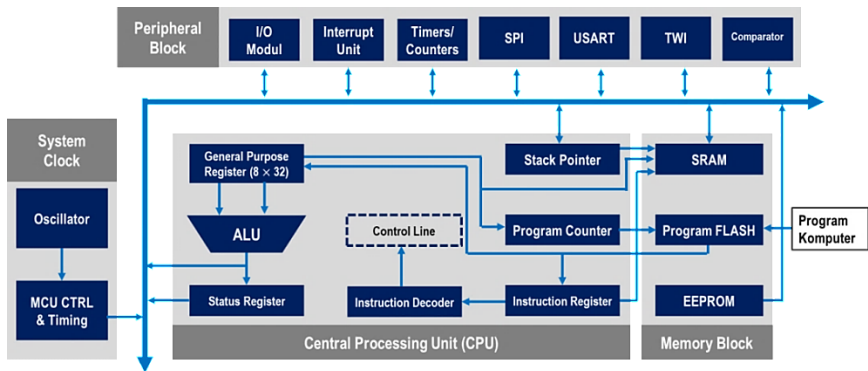
Pada dasarnya Mikrokontroler merupakan sebuah sistem Mikroprosesor lengkap berupa *chip* atau IC yang dapat diprogram. Mikrokontroler berbeda dengan Mikroprosesor serba guna yang digunakan dalam sebuah PC, karena di dalam sebuah Mikrokontroler umumnya juga telah berisi komponen pendukung sistem Mikroprosesor seperti inti prosesor, memori, dan antarmuka I/O. Sedangkan, di dalam Mikroprosesor umumnya hanya berisi CPU saja. Mikrokontroler populer digunakan pada berbagai keperluan otomatisasi seperti sistem kontrol mobil, perangkat medis, pengendali jarak jauh, mesin perkantoran, dan lain-lain. Penggunaan Mikrokontroler lebih ekonomis dibandingkan sebuah desain sistem yang berisikan Mikroprosesor, memori, dan perangkat input/output terpisah. Tabel 1.1 memperlihatkan perbedaan Mikrokontroler dengan Mikroprosesor secara umum.

Tabel 1.1 Perbedaan Mikrokontroler dengan Mikroprosesor

Mikrokontroler	Mikroprosesor
Mikrokontroler digunakan untuk mengeksekusi sebuah fungsi khusus	Mikroprosesor dapat mengeksekusi berbagai fungsi secara bersamaan
Desain dan <i>hardware</i> yang ekonomis	Desain dan <i>hardware</i> yang relatif mahal

Mikrokontroler	Mikroprosesor
Mudah diganti	Sulit diganti
Telah tertanam teknologi CMOS yang membutuhkan daya yang kecil untuk beroperasi	Membutuhkan konsumsi daya yang tinggi karena harus mengontrol keseluruhan sistem
Terdiri dari CPU, RAM, ROM, dan <i>port</i> I/O	Tidak tersusun atas RAM, ROM, dan <i>port</i> I/O. Mikroprosesor menggunakan pinnya untuk terhubung dengan perangkat periferal

Di pasaran saat ini telah tersedia berbagai jenis Mikrokontroler yang terus mengalami perkembangan. Salah satu jenis mikrokontroler yang banyak digunakan saat ini adalah Mikrokontroler AVR. AVR merupakan Mikrokontroler RISC (*Reduce Instruction Set Computing*) 8 bit berdasarkan arsitektur Harvard. Kelebihan Mikrokontroler jenis AVR bila dibandingkan dengan Mikrokontroler lain yaitu AVR memiliki kecepatan eksekusi program yang lebih cepat. Hal ini disebabkan karena sebagian besar instruksi dieksekusi dalam satu siklus *clock*. Selain itu, Mikrokontroler AVR memiliki fitur lengkap seperti ADC internal, EEPROM internal, *Timer/Counter*, PWM, *port* I/O, komunikasi serial, dan lain-lain. Arsitektur AVR secara garis besar terdiri dari empat diagram blok yaitu CPU, *memory block*, *system clock*, dan *peripheral block*. Gambar 1.1 memperlihatkan arsitektur ATmega16 yang merupakan salah satu *chip* AVR dari keluarga Atmel.



Gambar 1.1. Arsitektur AVR ATmega16

Fungsi dari masing-masing blok pada Gambar 1.1 adalah:

- a. **CPU** bertugas untuk memproses semua instruksi atau program yang ada di Mikrokontroler.
- b. **Memory Block** bertugas menyimpan data dan program. *Program flash* berfungsi untuk menyimpan kode program, sedangkan SRAM dan EEPROM berfungsi menyimpan data.
- c. **System Clock** bertugas menangani semua *clock* pada CPU, memori, ADC, input/output, dan lain-lain.
- d. **Peripheral Block** berfungsi untuk menghubungkan Mikrokontroler dengan *environment* di luar Mikrokontroler seperti I/O Modul, *Interrupt Unit*, *Timer/Counter*, SPI, USART, TWI, dan *Comparator*.

1.2 ARDUINO

Arduino merupakan *board* Mikrokontroler yang menggunakan *chip* AVR ATmega yang dirilis oleh Atmel dan bersifat *open source*. Saat ini Arduino menjadi salah satu proyek

perangkat keras *open source* yang paling populer. Sifat Arduino yang *open source* membuat Arduino berkembang dengan sangat cepat. Arduino banyak diminati karena kemudahan dalam penggunaan dan penulisan programnya. Jika pada *board* Mikrokontroler yang lain masih membutuhkan perangkat keras secara terpisah, tidak demikian dengan Arduino yang hanya membutuhkan kabel USB untuk dapat mulai digunakan. Selain itu, Arduino IDE yang merupakan software sekaligus *compiler* Arduino menggunakan bahasa pemrograman Bahasa C/C++ dengan versi yang lebih sederhana. *Compiler* yang digunakan Arduino pada dasarnya adalah *compiler* avr-gcc yang sama dengan yang digunakan avr-studio, namun *plugin* yang tersedia di Arduino memungkinkan siapa saja menambahkan *compiler* lain selain avr-gcc, bahkan dukungan untuk prosesor lain selain Atmel AVR.

Berbagai kemudahan serta fleksibilitas yang ditawarkan Arduino menarik antusias banyak orang untuk membangun kreativitas dengan membuat berbagai jenis project meski tidak memiliki latar belakang di bidang elektronika, teknik, maupun komputasi. Seperti Mikrokontroler kebanyakan, Arduino lahir dan berkembang kemudian muncul dengan berbagai tipe seperti:

a. Arduino Uno

Menggunakan *chip* ATmega328 sebagai Mikrokontrolernya, memiliki 14 pin I/O digital dan 6 input analog. Untuk pemrogramannya cukup menggunakan koneksi USB *type A to type B*.

b. Arduino Due

Tidak seperti kebanyakan Arduino, Arduino Due menggunakan *chip* yang lebih tinggi yaitu ARM Cortex CPU. Memiliki 54 pin I/O digital dan 12 pin input analog. Pemrogramannya menggunakan Micro USB.

c. Arduino Mega

Hampir mirip dengan Arduino Uno yang sama-sama menggunakan USB *type A to type B* namun menggunakan *chip* lebih tinggi berupa ATmega2560. Memiliki 54 pin I/O digital dan 16 pin input analog.

d. Arduino Leonardo

Menggunakan *chip* ATmega32U4 sebagai Mikrokontrolernya dengan jumlah pin I/O digital sebanyak 20 pin yang mana 7 diantaranya dapat digunakan sebagai PWM dan 12 pin sebagai input analog. Pemrogramannya membutuhkan kabel Micro USB.

e. Arduino Fio

Board Mikrokontroler ini memiliki jumlah pin I/O digital dan input analog yang sama dengan Arduino Uno. Akan tetapi, Arduino Fio memiliki Socket XBee yang memungkinkan Arduino tipe ini dapat dipakai untuk keperluan *project* yang berhubungan dengan *wireless*.

f. Arduino Lilypad

Menggunakan *chip* ATmega168 untuk versi lama dan ATmega328 untuk versi terbarunya. Memiliki 14 pin I/O digital dan 6 pin input analog. Arduino Lilypad berbentuk melingkar yang biasanya digunakan untuk produk pakaian dan tekstil.

g. Arduino Nano

Sudah dilengkapi dengan FTDI untuk pemrograman melalui Micro USB. Arduino tipe Nano menggunakan *chip* ATmega168 dan juga ada yang menggunakan ATmega328. Memiliki 14 pin I/O digital dan 8 pin input analog.

h. Arduino Mini

Arduino Mini memiliki fasilitas yang sama dengan Arduino Nano, hanya saja tidak dilengkapi dengan Micro USB untuk pemrogramannya.

i. Arduino Micro

Berbasis *chip* ATmega32U4 dengan 20 pin I/O digital dan 12 pin input analog.

j. Arduino Ethernet

Arduino tipe ini sudah dilengkapi dengan fasilitas Ethernet yang memungkinkan untuk pembuatan *project* yang berhubungan dengan jaringan LAN pada komputer. Adapun fasilitas pin I/O yang tersedia sama dengan Arduino Uno.

k. Arduino Esplora

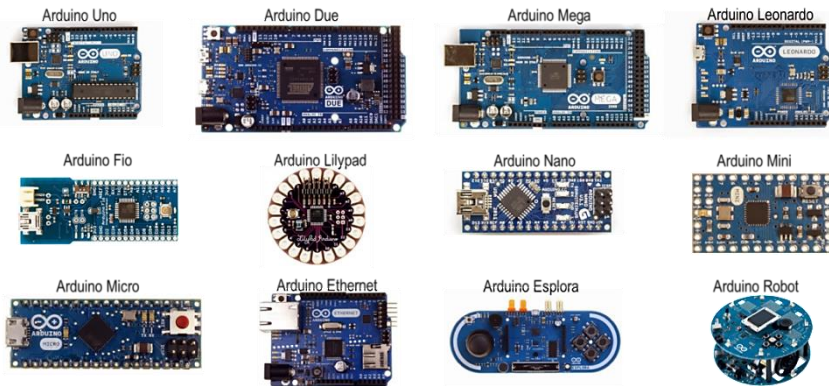
Arduino tipe ini sudah dilengkapi dengan beberapa sensor seperti sensor suhu, sensor *accelerometer*, dan sensor cahaya. Selain itu, Arduino ini dilengkapi dengan tombol *joystick*, *linear potentiometer*, 4 *push button*, LED, dan *microphone*.

l. Arduino Robot

Arduino tipe ini merupakan paket komplit Arduino yang sudah berbentuk robot dengan dilengkapi LCD, *speaker*, roda, sensor inframerah, dan lain-lain.

Selain tipe-tipe Arduino di atas, masih terdapat berbagai tipe Arduino lain seperti Arduino BT, Arduino

Duemilanove, Arduino Nouve Generazione, Arduino Extreme, dan lain-lain. Gambar 1.2 memperlihatkan tampilan fisik beberapa jenis Arduino yang banyak digunakan.



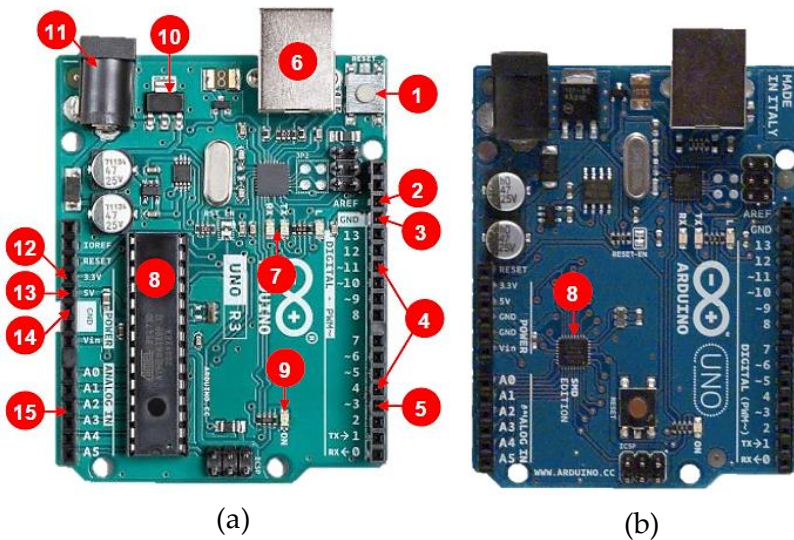
Gambar 1.2. Beberapa tipe *board* Arduino

Dari beberapa tipe *board* Arduino yang telah diuraikan di atas, Arduino Uno merupakan salah satu tipe Arduino yang paling sering digunakan. Arduino jenis ini menjadi pilihan banyak orang termasuk pemula karena ketersediaan referensi yang cukup melimpah baik di buku maupun internet. Selain itu, Arduino Uno memiliki jumlah pin I/O yang sudah mencukupi untuk berbagai keperluan pembuatan *project*. Penggunaan Arduino Uno dapat menjadi pondasi awal yang tepat sebelum menggunakan jenis Arduino yang lebih *advance* seperti Arduino Robot, Arduino Ethernet, Arduino Esplora dan yang lainnya. Berikut adalah spesifikasi dari Arduino Uno:

- Mikrokontroler : ATmega328P
- Tegangan Operasi : 5 V
- Tegangan Input : 7-12V (Rekomendasi)

- Pin I/O Digital : 14 pin (6 pin PWM)
- Pin Input Analog : 6 *channel*
- Arus DC Tiap Pin I/O : 20 mA
- Arus DC pada Pin 3.3V : 50 mA
- *Flash Memory* : 32 KB (ATmega328)
- SRAM : 2 KB (ATmega328)
- EEPROM : 1 KB (ATmega328)
- Clock Speed : 16 MHz

Berdasarkan dimensinya, *chip* ATmega328P pada Arduino Uno terbagi atas tipe DIP (*Dual Inline Package*) dan SMD (*Surface Mount Device*). Gambar 1.3 memperlihatkan *board* Arduino dengan *chip* bertipe DIP dan SMD.



Gambar 1.3. Arduino dengan *chip* (a) tipe DIP dan (b) tipe SMD

Bagian-bagian utama pada Arduino Uno berdasarkan Gambar 1.3 adalah sebagai berikut:

- 1) **Reset.** Fungsi tombol ini bukan untuk menghapus program melainkan menjalankan ulang program dari awal. Untuk mereset program juga dapat dilakukan dengan menghubungkan pin reset dengan *ground* secara singkat.
- 2) **AREF.** Merupakan referensi tegangan untuk input analog.
- 3) **GND.** *Ground* atau pin negatif dalam sirkuit elektronik.
- 4) **I/O Digital.** Arduino Uno memiliki 14 pin I/O digital yang tersedia pada pin 0 - 13 yang berfungsi memberikan nilai logika 0 atau 1 atau dapat berfungsi layaknya saklar. Logika 1 bernilai 5 V, sementara logika 0 bernilai 0 V. Pin berlabel “~” berfungsi sebagai PWM.
- 5) **PWM.** Menyediakan output PWM sebesar 8-bit (0-255) pada pin 3, 5, 6, 9, 10, dan 11.
- 6) **Power USB.** Berfungsi menghubungkan *board* Arduino ke komputer melalui koneksi USB sebagai suplai listrik ke *board* atau untuk pemrograman Arduino.
- 7) **TX/RX (Transmit/Receive).** LED TX/RX berkedip saat pemrograman di *chip* atau *board* Arduino berlangsung.
- 8) **ATmega328P.** Merupakan pusat kontrol Arduino. *Chip* ini diprogram oleh *board* Arduino melalui *software* Arduino IDE.
- 9) **LED Indikator Power.** Akan menyala saat *board* Arduino diberikan suplai listrik.
- 10) **Voltage Regulator.** IC ini berfungsi untuk mengatur/menstabilkan tegangan eksternal yang terhubung melalui *power jack*.

- 11) **Power Jack (DC).** Untuk koneksi Arduino dengan sumber listrik dengan input DC sebesar 5 - 12V.
- 12) **Pin 3,3V.** Sumber tegangan output 3,3 V.
- 13) **Pin 5V.** Sumber tegangan output 5 V.
- 14) **GND.** Ground atau pin negatif dalam sirkuit elektronik.
- 15) **Pin Analog.** Arduino Uno memiliki 6 *channel* input analog yang tersedia pada pin A0 - A5. Pin ini berfungsi menerima input analog dari berbagai komponen seperti sensor untuk selanjutnya diubah menjadi bentuk digital menggunakan ADC.

1.3 ARDUINO IDE

Arduino IDE merupakan *software* pemrograman pada *board* Arduino yang menggunakan bahasa C/C++ namun dengan versi yang sudah disederhanakan. IDE merupakan singkatan dari *Integrated Development Environment* yang secara sederhana dapat diartikan sebagai lingkungan terintegrasi yang digunakan untuk keperluan pengembangan program. Disebut sebagai *environment*/lingkungan karena melalui *software* Arduino IDE tersebut dilakukan pemrograman agar dapat melakukan fungsi-fungsi yang dibenamkan melalui *syntax* pemrograman yang telah tersedia. Kode program pada Arduino IDE biasa disebut *sketch* yang jika sudah selesai dibuat pada Arduino IDE dapat langsung di-*compile* dan di-*upload* ke *board* Arduino.

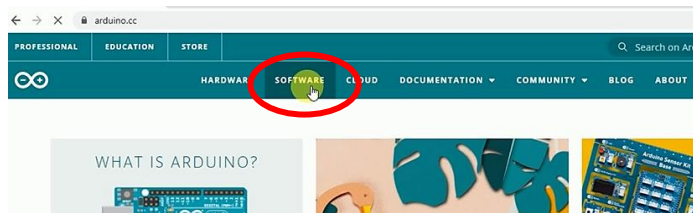
a. Proses *Download* dan Instalasi Arduino IDE

Berikut adalah *step by step* tahapan *download* dan instalasi Arduino IDE menggunakan sistem operasi Windows. Pada

sistem operasi lain seperti Linux dan Mac OS juga tidak jauh berbeda.

1) Tahapan *Download*

- *Software* Arduino bersifat *free access* dan dapat di-download di <https://www.arduino.cc>
- Pada bagian menu pilih *software*



Gambar 1.4. Menu pada arduino.cc

- Pada *download options*, pilih *installer* sesuai sistem operasi yang digunakan, dalam hal ini misalkan menggunakan sistem operasi Windows.



Gambar 1.5. Opsi *download*

- Jika sudah muncul opsi *download* seperti Gambar 1.6, klik *Just Download* atau *Contribute & Download* jika ingin berkontribusi. Setelah itu, proses *download*

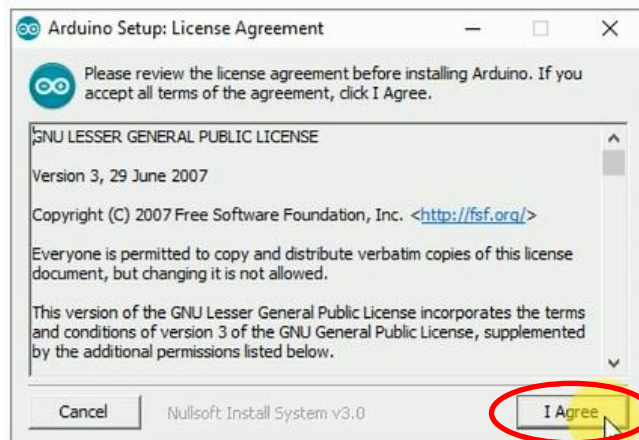
akan berlangsung. Tunggu beberapa saat hingga *file* ter-download dengan sempurna.



Gambar 1.6. Memulai proses *download*

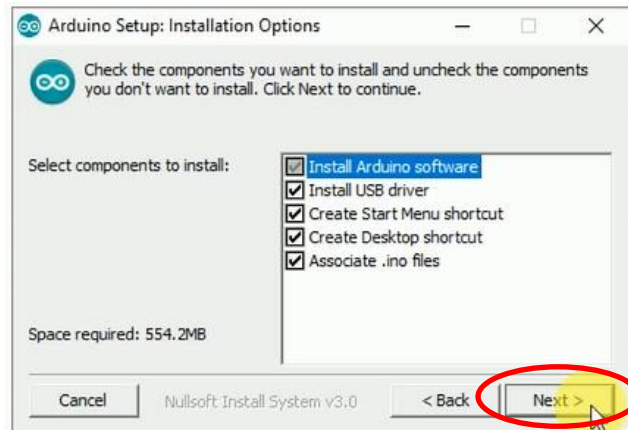
2) Tahapan Instalasi

- Double klik atau *run as administrator* pada file *installer* yang telah diunduh. Tunggu beberapa saat hingga muncul *License Agreement*. Klik *I Agree*.



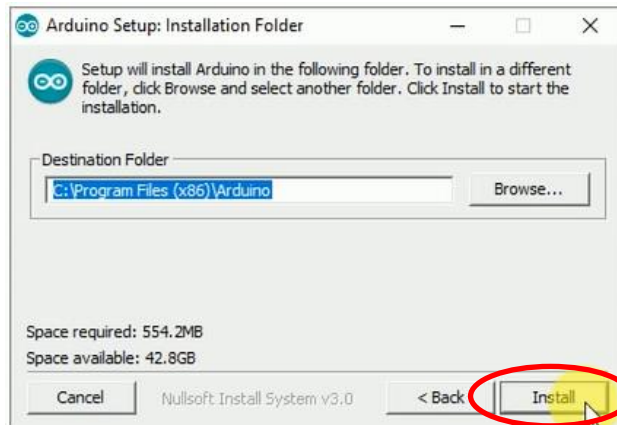
Gambar 1.7. *License Agreement*

- Untuk *Installation Option* centang semua opsi dan klik tombol *Next*.



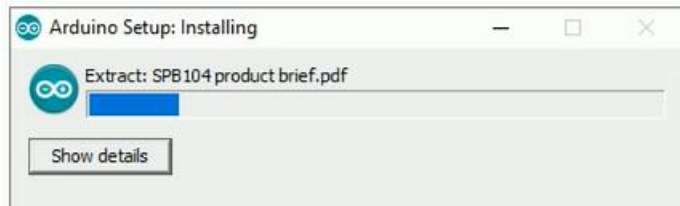
Gambar 1.8. *Installation Options*

- Pada **Installation Folder** dapat dipilih folder apa yang akan digunakan sebagai tempat menyimpan program. Selanjutnya klik **Install**.



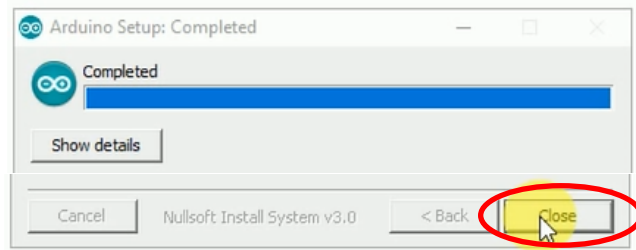
Gambar 1.9. *Installation Folder*

- Proses ekstraksi dan instalasi dimulai. Proses ini biasanya memakan waktu beberapa menit.



Gambar 1.10. Proses ekstraksi dan instalasi

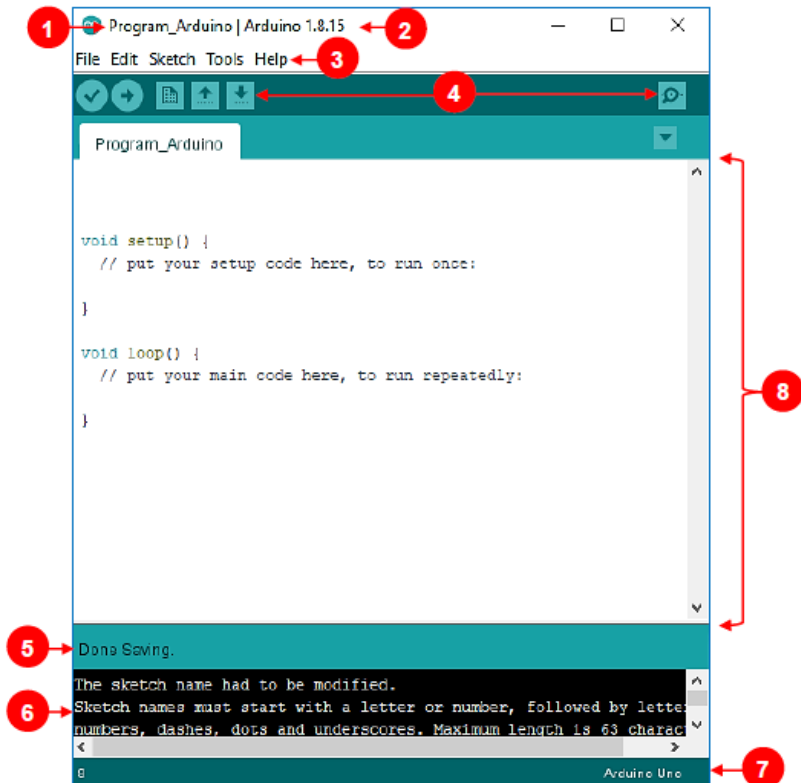
- Tunggu beberapa saat sampai proses instalasi selesai kemudian klik *close*. *Software* Arduino IDE siap digunakan.



Gambar 1.11. Proses instalasi selesai

b. Jendela *Software* Arduino IDE

Jendela *software* Arduino IDE terdiri dari beberapa bagian seperti menu, tombol *toolbar*, *text editor* untuk penulisan *sketch*, dan *console* sebagai informasi umpan balik. Gambar 1.11 memperlihatkan tampilan jendela *software* Arduino IDE.



Gambar 1.12. Jendela *software* Arduino IDE


Bagian-bagian pada *software* Arduino IDE berdasarkan Gambar 1.11 di atas adalah sebagai berikut:

- 1) **Nama file:** File disimpan dengan ekstensi file .ino.
- 2) **Versi Arduino IDE:** *Software* Arduino IDE juga memiliki versi yang bersifat *portable* yang dapat digunakan tanpa melalui tahapan instalasi terlebih dahulu.
- 3) **Menu**
 - **File:** Berisi beberapa opsi untuk membuat *file* baru (**new**), membuka file yang telah disimpan (**open**),






menggunakan contoh *sketch* yang tersimpan pada Arduino IDE (**examples**), menyimpan program (**save, save as**), dan lain-lain.

- **Edit:** Berisi opsi untuk membatalkan perintah sebelumnya dan sebaliknya (**Undo, Redo**), memperbesar atau memperkecil huruf (**Increase-Decrease Font Size**), menyalin *sketch* sebagai teks HTML, dan berbagai opsi lainnya.
- **Sketch:** Dapat digunakan untuk memverifikasi dan melakukan *compile* program (**Verify/Compile**), meng-*upload* program, memasukkan *library* baru, dan masih banyak lagi termasuk opsi untuk mengubah bentuk *file* biner (**Export Compiled Binary**).
- **Tools:** Menyediakan fasilitas untuk mengatur format penulisan program agar lebih rapi (**Auto Format**), mengatur *library* (**Manage Libraries**), menampilkan jendela serial (**Serial Monitor**), juga untuk mengatur penggunaan *board* dan *port* yang harus diatur saat menggunakan Arduino (**Board, Port**).
- **Help:** Menyediakan informasi seputar *troubleshooting*, informasi serta pertanyaan yang sering muncul tentang Arduino (**About Arduino, FAQ**), dan tersedia juga link situs lengkap tentang Arduino (**Visit Arduino.cc**).

4) Toolbar

- **Verify **: Pada versi lama dikenal dengan **Compile**. Tombol ini berfungsi untuk memverifikasi *sketch* yang dibuat apakah sudah sesuai atau masih terdapat kesalahan. Proses verifikasi diperlukan

untuk mengubah *sketch* ke bentuk *binary code* sebelum di-*upload* ke Mikrokontroler.

- **Upload**  : Berfungsi untuk meng-*upload* *sketch* ke *board* Arduino. Jika program belum diverifikasi menggunakan tombol **Verify**, maka tombol *upload* ini akan meng-*compile sketch* dan langsung di-*upload* ke *board* Arduino.
 - **New Sketch**  : Digunakan untuk membuat *window* dan *sketch* baru.
 - **Open Sketch**  : Berfungsi untuk membuka *sketch* yang pernah dibuat dengan ekstensi file *.ino*.
 - **Save Sketch**  : Berfungsi untuk menyimpan file.
 - **Serial Monitor**  : Digunakan untuk membuka *interface* komunikasi serial.
- 5) **Keterangan Aplikasi:** Notifikasi yang dikerjakan oleh *software* seperti “**Compiling**” dan “**Uploading**” saat memverifikasi dan meng-*upload* program ke Arduino akan muncul pada bagian ini.
 - 6) **Console:** Rincian pesan-pesan yang dikerjakan *software* beserta pesan-pesan tentang *sketch* akan muncul pada bagian ini. Contohnya jika terjadi kesalahan penulisan program, maka akan muncul informasi *error*. Hal ini memudahkan pengguna dalam memperbaiki program yang telah dibuat.
 - 7) **Informasi Board dan Port:** Tipe board Arduino dan *port* yang digunakan akan ditampilkan pada bagian ini.
 - 8) **Tempat Penulisan Program / Sketch:** Semua kegiatan pemrograman pada Arduino dilakukan pada bagian ini.

c. *Sketch* Arduino IDE

Struktur *sketch* pada Arduino IDE dikelompokkan menjadi 3 blok utama yaitu **Header**, **Setup**, dan **Loop**. Namun, pada program yang lebih kompleks biasanya dibuat blok-blok lain untuk menyediakan fungsi-fungsi pendukung.

1) **Header**

Bagian ini digunakan sebagai tempat untuk mendeskripsikan atau mendeklarasikan variabel tertentu yang nantinya akan digunakan pada program utama. Selain itu, *library* juga dapat dimasukkan melalui bagian ini. Program yang ditulis pada bagian header hanya dieksekusi sekali saja yaitu pada saat proses *compile*. Berikut contoh kode program untuk memasukkan *library* sensor temperatur LM35 dan LED yang terhubung dengan pin 2.

```
#include < LM35.h>
int Led = 2;
```

2) **Setup**

Bagian ini dijalankan sekali saja yaitu pada saat program baru dieksekusi atau saat program direset. Bagian ini dapat digunakan untuk mengatur mode pin (input atau output), mengatur *baudrate* (kecepatan transfer data), mengatur *timer*, dan lain-lain. Deklarasi /inisialisasi variabel juga dapat dilakukan pada bagian ini. Berikut adalah contoh penulisan kode program pada bagian *setup* berupa pengaturan mode pin dan *baudrate*.

```
void setup()
{
  pinMode (Led, OUTPUT);
  Serial.begin(9600);
}
```

3) Loop

Bagian loop merupakan fungsi utama program yang akan dijalankan secara berulang-ulang dan akan berhenti ketika board Arduino sudah tidak menerima daya / power. Berikut adalah contoh kode program untuk menghidupkan dan mematikan LED secara berulang-ulang dengan delay 1 detik.

```
void loop()
{
  digitalWrite (Led, HIGH);
  delay (1000);
  digitalWrite (Led, LOW);
  delay (1000);
}
```

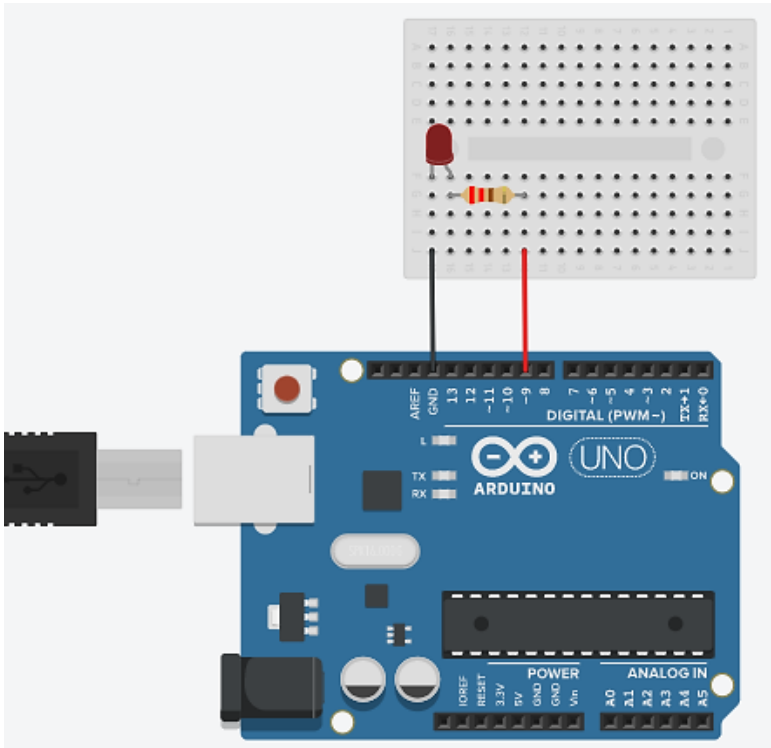
1.4 TAHAPAN PERCOBAAN MENGGUNAKAN ARDUINO

Pada bagian ini akan dibahas mengenai prosedur melakukan percobaan sederhana berupa *blinking LED* (LED berkedip) menggunakan *board* Arduino Uno dengan Arduino IDE sebagai media pembuatan programnya.

- a. Siapkan alat dan bahan, berupa:
 - 1) 1 unit Arduino Uno beserta kabel USB *type A to type B*
 - 2) 1 buah LED
 - 3) 1 buah resistor 220 Ohm
 - 4) 2 buah kabel *male to male*

- 5) 1 buah papan rangkaian
- 6) 1 unit PC yang sudah terinstal Arduino IDE.

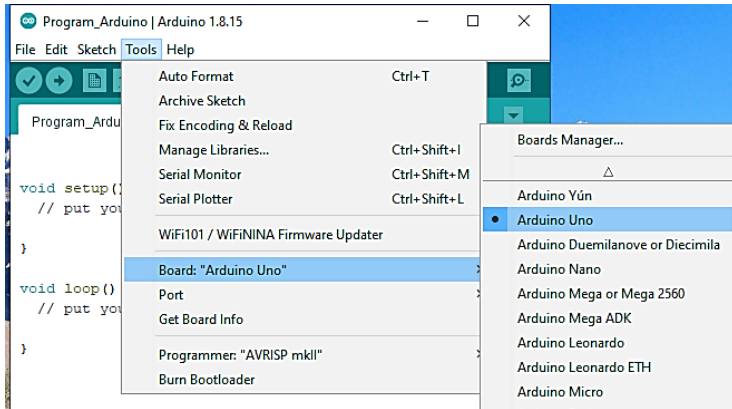
b. Buatlah rangkaian seperti Gambar 1.13 berikut:



Gambar 1.13. Contoh rangkaian LED sederhana

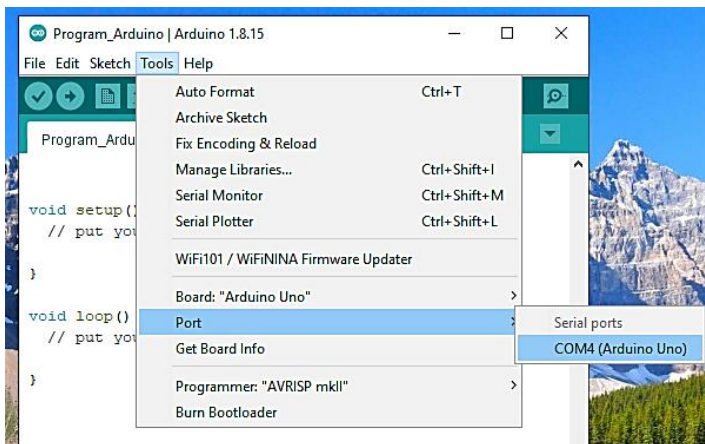
- c. Hubungkan *board* Arduino ke PC melalui kabel USB *type A to type B*. Pastikan LED Power pada *board* Arduino menyala untuk memastikan bahwa *board* sudah terhubung dengan PC.
- d. Buka *software* Arduino IDE yang sudah terinstal di PC.

- e. Pilih *board* Arduino yang digunakan, dalam hal ini adalah Arduino Uno dengan mengakses **Tools > Board > Arduino Uno**.



Gambar 1.14. Memilih *board* Arduino

- f. Pilih *port* yang akan digunakan melalui **Tools > Port**. Biasanya Arduino IDE akan mendeteksi *port* secara otomatis jika *board* Arduino dihubungkan ke PC.



Gambar 1.15. Memilih *Port*

- g. Jika sudah mengatur *board* dan *port*, selanjutnya ketik *sketch* berikut di Arduino IDE.

```
int Led = 9;

void setup() {
  pinMode (Led, OUTPUT);
}

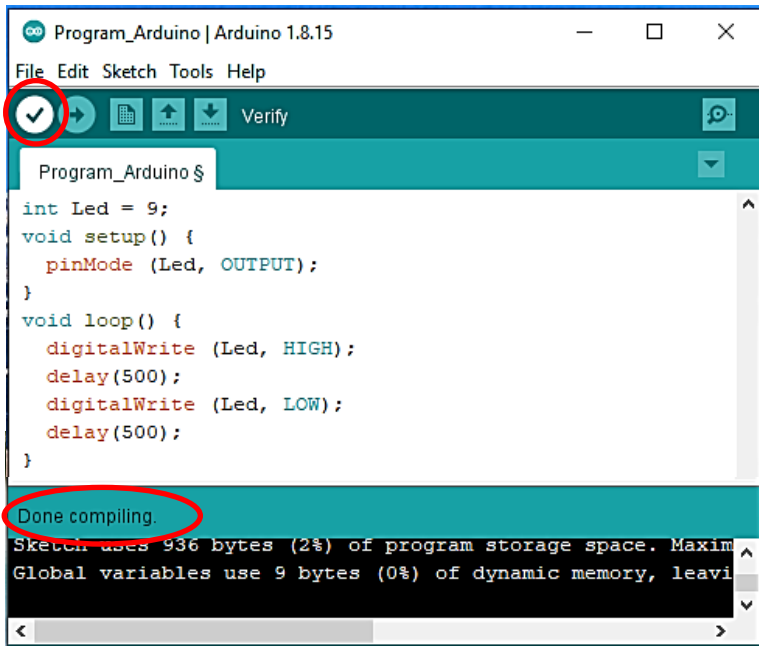
void loop() {
  digitalWrite (Led, HIGH);
  delay (500);
  digitalWrite (Led, LOW);
  delay (500);
}
```



Berikut penjelasan masing-masing blok pada *sketch*.

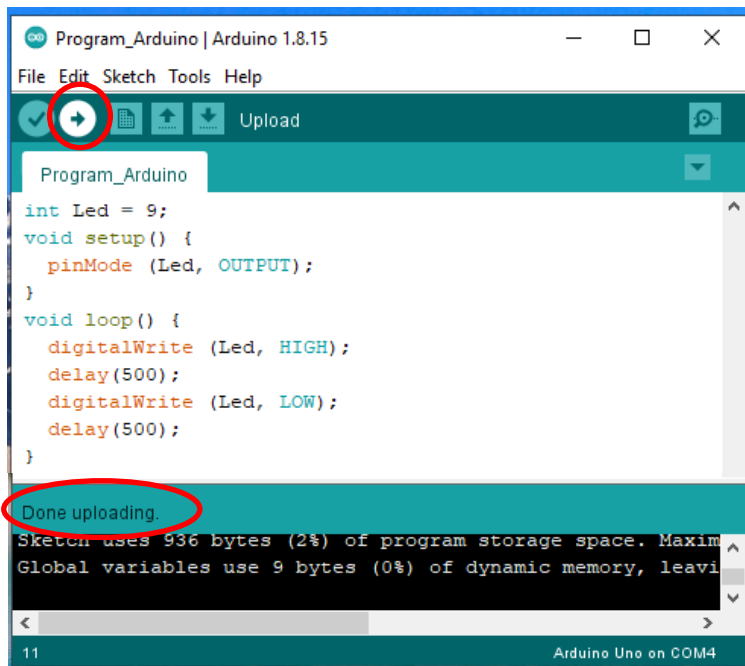
- Header. Berisi inisialisasi variabel **Led** dengan tipe data angka satuan (integer) bernilai 9, artinya pin 9 pada Arduino Uno dijadikan sebagai input/output.
- Setup: Kode **pinMode(Led,OUTPUT)** berfungsi untuk memberitahu Arduino bahwa pin 9 akan dijadikan sebagai output. Kode ini dipanggil hanya sekali.
- Loop: Bagian ini berisi program utama yang dipanggil dan dijalankan berulang kali. Kode **digitalWrite (Led,HIGH)** berarti LED akan mendapat logika HIGH atau 1 yang berarti menerima tegangan 5V, sehingga LED menyala. Sedangkan, **digitalWrite (Led,LOW)** berarti LED mendapat logika LOW atau 0 yang berarti menerima tegangan 0V dan menyebabkan LED padam. LED akan menyala dan padam secara terus menerus dengan delay 500ms sesuai kode **delay(500)**.

- h. Simpan dan *compile* program dengan mengklik **Verify**. Tunggu beberapa saat sampai muncul keterangan **Done compiling**.



Gambar 1.16. Proses *compile* program

- i. Upload program ke *board* Mikrokontroler dengan mengklik tombol **Upload**. Proses upload mulai berjalan biasanya ditandai dengan berkedipnya LED TX/RX pada *board* Arduino. Tunggu sampai muncul keterangan **Done uploading**.



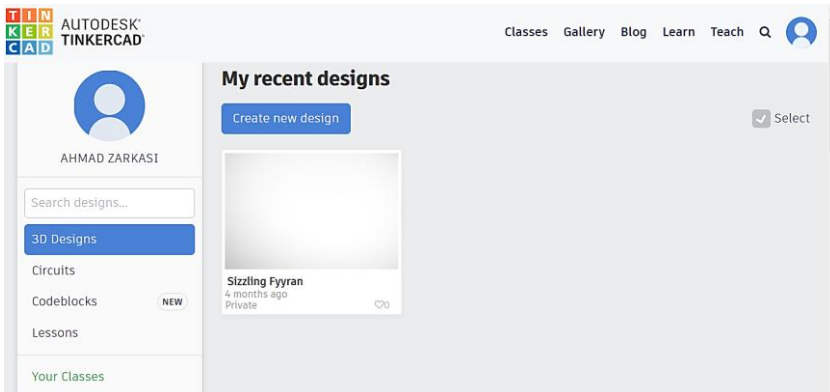
Gambar 1.17. Proses *upload* program

BAB II

PLATFORM TINKERCAD

2.1 PENGENALAN TINKERCAD

Tinkercad merupakan platform desain berbasis *website* yang menyediakan berbagai sarana untuk kebutuhan desain seperti desain 3D, codeblock, dan rangkaian elektronika. *Platform* ini dirilis oleh Autodesk sudah cukup lama yaitu pada tahun 2011. Meski demikian, pengembangannya terus berlanjut hingga saat ini. Salah satu kelebihan *platform* berbasis *website* seperti Tinkercad adalah mudah diakses oleh siapapun tanpa perlu melakukan instalasi software di PC. Tak hanya itu, Tinkercad juga dapat digunakan melalui *smartphone* dan tablet sehingga menjadi lebih fleksibel dan praktis. Gambar 2.1 memperlihatkan tampilan awal web Tinkercad.

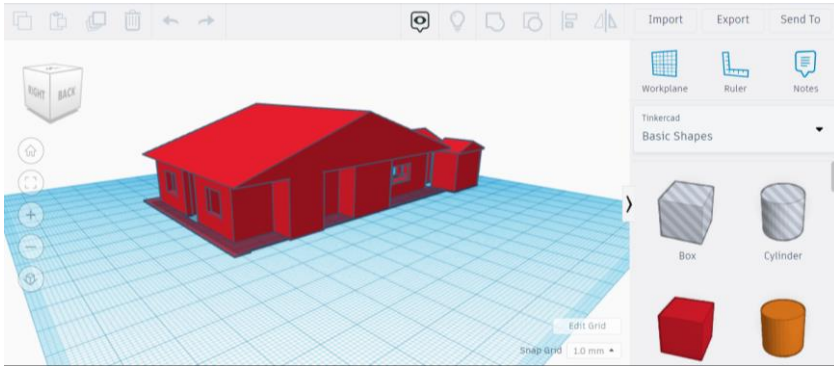


Gambar 2.1. Tampilan awal web Tinkercad

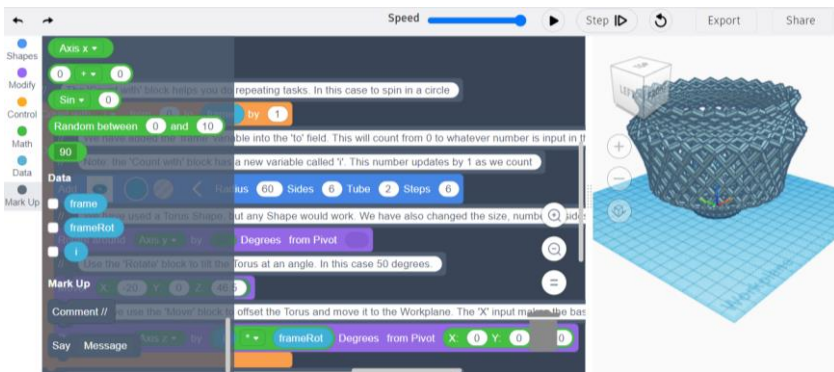
Tinkercad pada dasarnya merupakan *platform* yang bersifat *open source* sehingga siapa saja dapat mengembangkannya dengan mudah. Web yang dimiliki Tinkercad juga dapat dijadikan media pembelajaran secara daring (*online*) karena hampir sama dengan web penyedia layanan pembelajaran *online* seperti Edmodo, Google Classroom, dan Schoology. Hanya saja, Tinkercad lebih dikhususkan untuk kebutuhan desain. Pada web Tinkercad tersedia menu **Classes** yang dapat digunakan oleh pendidik untuk memantau hasil kerja siswa/mahasiswa seperti yang terlihat pada Gambar 2.1. Siswa/mahasiswa yang ingin mengikuti pembelajaran melalui Tinkercad bahkan tidak harus membuat akun atau melakukan registrasi. Cukup dengan memasukkan kode unik serta *nickname* yang diberikan *educator*, maka siswa/mahasiswa bersangkutan akan langsung dapat bergabung.

Secara umum, menu desain yang disediakan oleh Tinkercad terdiri dari 3 menu utama yaitu **3D Designs**, **Codeblocks**, dan **Circuits**. Penggunaan menu **3D Designs** dan **Codeblocks** pada dasarnya sama yaitu membuat objek 3 dimensi. Perbedaannya terletak pada proses dan tujuan pembuatannya. Pembuatan objek menggunakan **3D Designs** dilakukan dengan memanfaatkan *shapes* yang tersedia di Tinkercad dengan berbagai bentuk yang dapat diatur langsung. Sedangkan, pada **Codeblocks**, pembuatan objek dilakukan dengan menyusun kode yang tersedia pada blok-blok di Tinkercad seperti menyusun sebuah kode program. Kelebihan menggunakan Codeblocks ini adalah dapat

disimulasikan setiap tahapan pembuatannya. Gambar 2.2 memperlihatkan perbedaan **3D Designs** dan **Codeblocks**.



(a)

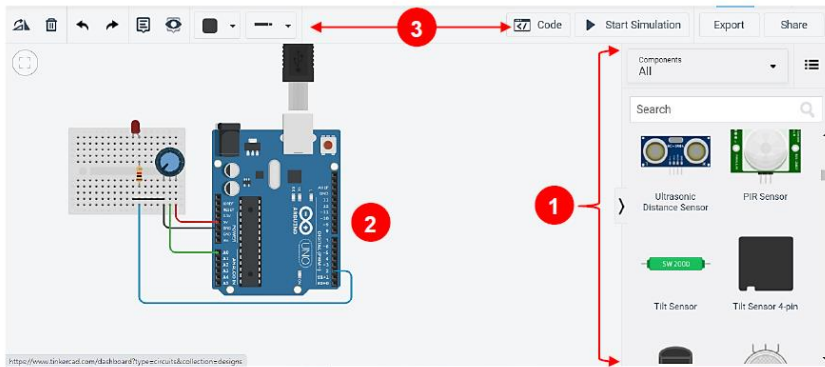


(b)

Gambar 2.2. Contoh desain 3D menggunakan (a) 3D Designs dan (b) Codeblocks

Selain menyediakan menu untuk kebutuhan desain objek 3 dimensi, Tinkercad juga menyediakan menu **Circuits** untuk melakukan simulasi rangkaian elektronika, yang mana akan menjadi fokus pada buku ini. Menu **Circuits** dapat dijadikan sebagai media simulasi Mikrokontroler karena di

dalamnya sudah tersedia Arduino Uno yang dapat diprogram menggunakan kode program yang sama persis dengan Arduino IDE. Tak hanya itu, Tinkercad juga telah menyediakan *serial monitor* untuk membaca nilai serial. Contoh tampilan pada menu Circuits dapat dilihat pada Gambar 2.3.



Gambar 2.3. Contoh rangkaian sederhana di menu Circuits

Bagian-bagian pada Gambar 2.3 secara garis besar dapat dijelaskan sebagai berikut:

1) **Components**

Bagian ini berisi beberapa komponen elektronika yang dapat diakses melalui proses *drag and drop*.

2) **Tempat Membuat Rangkaian**

Merupakan *space* yang disediakan Tinkercad untuk membuat rangkaian dengan menggunakan komponen yang tersedia. Rangkaian yang telah dibuat dapat digeser atau di-*zoom* menggunakan *mouse*.

3) Toolbar

- **Rotate** 


Berfungsi memutar komponen searah jarum jam

- **Delete** 

Berfungsi menghapus komponen

- **Undo/Redo** 

Berfungsi untuk membatalkan eksekusi/perintah saat pembuatan rangkaian dan sebaliknya.

- **Notes tool** 

Digunakan untuk membuat catatan pada komponen atau rangkaian.

- **Toggle notes visibility** 

Berfungsi untuk menampilkan atau tidak menampilkan notes.

- **Wire color** 

Berfungsi mengatur warna kabel/kawat

- **Wire type** 

Berfungsi mengatur jenis kabel/kawat

- **Toggle code editor** 

Berfungsi menyediakan area penulisan *sketch* seperti yang dimiliki oleh Arduino IDE. Selain itu juga disediakan serial monitor untuk melihat data serial dari rangkaian.

- **Start/stop simulation** 

Berfungsi memulai dan menghentikan simulasi rangkaian. Tombol ini akan berfungsi hanya jika sketch yang dibuat sudah benar, artinya dapat digunakan sebagai tombol verifikasi seperti pada Arduino IDE.

- **Export file**

Export

Digunakan jika ingin membuat PCB dari rangkaian. Ekstensi file yang dihasilkan adalah .brd yang dapat dibuka melalui *software* Eagle.

- **Share this design**

Share

Project simulasi yang telah dibuat dapat dibagikan melalui link tertentu.

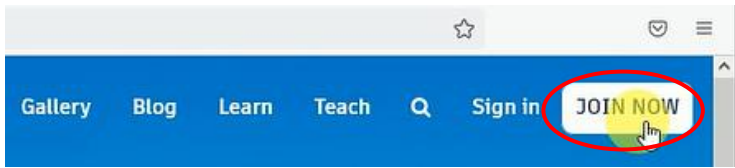
2.2 REGISTRASI, PEMBUATAN CLASS, DAN JOIN CLASS DI TINKERCAD

Sebelum menggunakan Tinkercad, para pengguna harus mengetahui bagaimana cara masuk ke *platform* Tinkercad baik melalui tahapan registrasi ataupun melalui **class**.

a. Registrasi / Pembuatan Akun di Tinkercad

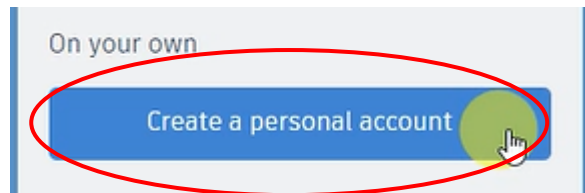
Tahapan registrasi penting untuk diketahui oleh mahasiswa, para pendidik (guru/dosen), asisten praktikum, ketua kelompok *project* tertentu, atau siapa saja yang ingin memanfaatkan Tinkercad secara pribadi. Karena ketika seorang pengguna sudah memiliki akun, maka ia dapat menentukan apakah ingin menjadi *students*, *teacher/educator*, atau yang lainnya sesuai kebutuhan. Berikut akan dijelaskan tahapan registrasi atau pembuatan akun di Tinkercad.

- 1) *Platform* Tinker bersifat *open source* yang dapat diakses melalui <https://tinkercad.com/>
- 2) Pada bagian menu pilih **Join Now**.



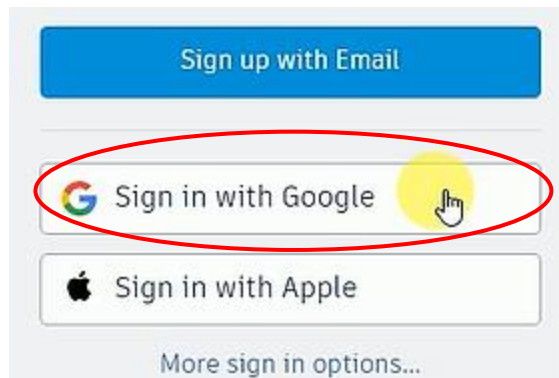
Gambar 2.4. Menu pada tampilan awal Tinkercad

- 3) Pada pilihan **Start Tinkering**, pilih **Create a personal account**.



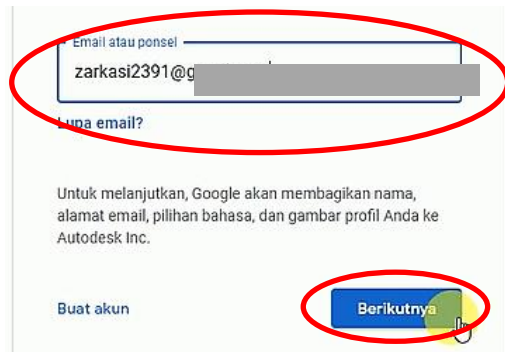
Gambar 2.5. Create a personal account di Tinkercad

- 4) Pilih metode pendaftaran yang diinginkan. Contoh jika ingin mendaftar menggunakan akun gmail, maka pilih **Sign in with Google**.



Gambar 2.6. Pilihan mode untuk pendaftaran Tinkercad

- 5) Masukkan alamat email, lalu klik **Berikutnya**.

A screenshot of a registration form. At the top, there is a text input field labeled "Email atau ponsel" containing the text "zarkasi2391@g...". Below this field is a link that says "Lupa email?". Further down, a paragraph of text states: "Untuk melanjutkan, Google akan membagikan nama, alamat email, pilihan bahasa, dan gambar profil Anda ke Autodesk Inc." At the bottom left is a link "Buat akun". At the bottom right is a blue button labeled "Berikutnya" with a yellow highlight and a hand cursor icon pointing at it. Red circles are drawn around the email input field and the "Berikutnya" button.

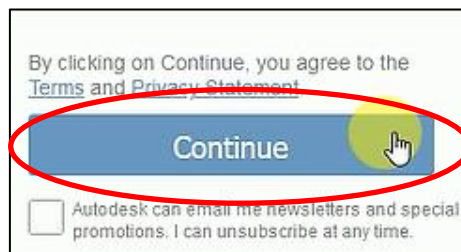
Gambar 2.7. Memasukkan alamat email

- 6) Masukkan *password* kemudian tekan **Berikutnya**.

A screenshot of a registration form. It features a text input field labeled "Masukkan sandi Anda" filled with black dots. Below the field is a checkbox labeled "Tampilkan sandi". To the left of the checkbox is a link "Lupa sandi?". To the right is a blue button labeled "Berikutnya" with a yellow highlight and a hand cursor icon pointing at it. Red circles are drawn around the password input field and the "Berikutnya" button.

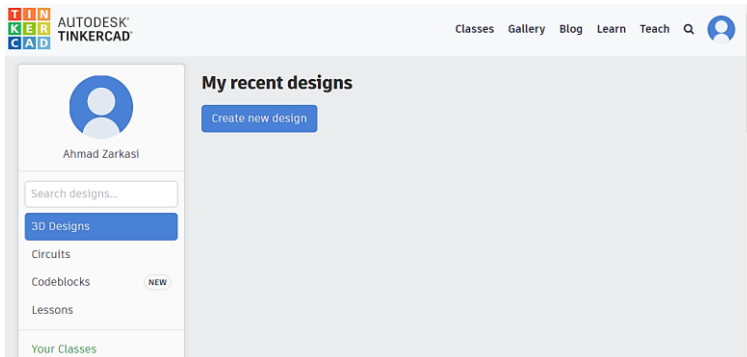
Gambar 2.8. Memasukkan *password*

- 7) Jika muncul tampilan seperti ini, klik **Continue**.

A screenshot of a confirmation screen. At the top, it says "By clicking on Continue, you agree to the [Terms and Privacy Statement](#)". Below this is a large blue button labeled "Continue" with a yellow highlight and a hand cursor icon pointing at it. At the bottom, there is a checkbox followed by the text: "Autodesk can email me newsletters and special promotions. I can unsubscribe at any time." A red circle is drawn around the "Continue" button.

Gambar 2.9. Konfirmasi promosi dari Autodesk

- 8) Jika sudah berhasil maka akan muncul tampilan seperti ini.



Gambar 2.10. Tampilan awal pada akun Tinkercad

Setelah melakukan registrasi, maka untuk masuk ke akun Tinkercad dapat dilakukan dengan mengikuti langkah 1) sampai 4) atau bisa juga melalui menu **sign in**.

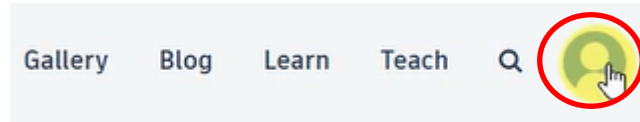
b. Pembuatan Classes

Tinkercad menyediakan fitur berupa **Classes** / **Class** yang dapat dimanfaatkan untuk kegiatan pembelajaran secara *online* layaknya Google Classroom, Edmodo, dan Schoology. Fitur ini dapat dimanfaatkan untuk keperluan penugasan yang dilakukan di Tinkercad baik oleh dosen, guru, asisten praktikum, atau mungkin ketua kelompok suatu *project*. Seorang ketua kelompok dapat dengan mudah melakukan monitoring hasil pekerjaan anggotanya melalui akun yang ia miliki. Penggunaan fitur Classes juga memungkinkan anggota tim untuk dapat mengerjakan tugas atau membuat *project* tanpa perlu membuat akun di Tinkercad. Cukup dengan **class code** dan **nickname** yang

dikirimkan oleh ketua kelompok, anggota tim dapat dengan mudah mengakses Tinkercad. Berikut tahapan pembuatan class di Tinkercad.

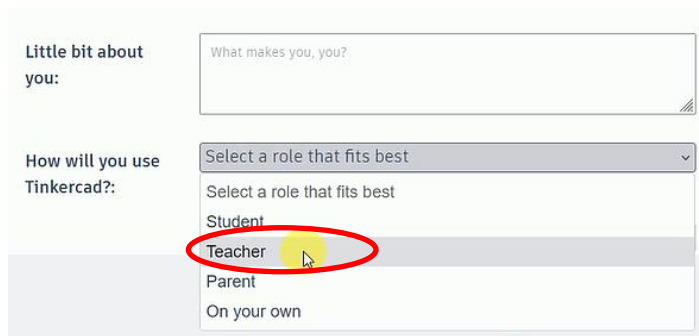
1) Ubah status menjadi **Teacher**, dengan cara:

- Masuk ke akun Tinkercad. Kemudian klik profil.



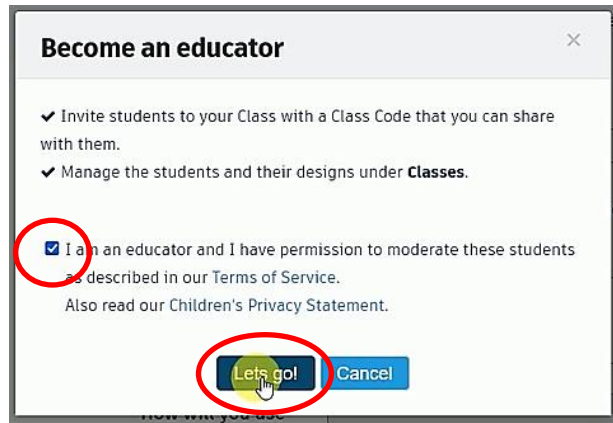
Gambar 2.11. Klik profil

- Pada **Account Settings** di bagian paling bawah, klik pilihan **Select a role that fits best**, kemudian pilih **Teacher**.



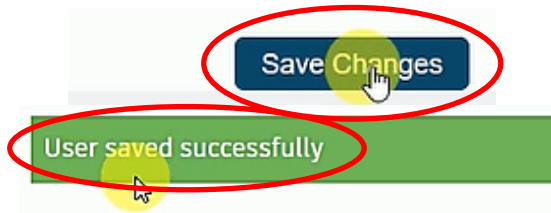
Gambar 2.12. Select role

- Centang kotak **Become an educator**, kemudian klik **Lets go!** (Gambar 2.13).



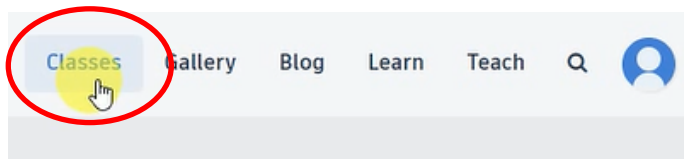
Gambar 2.13. Become an educator

- Klik **Save changes** sampai muncul keterangan **User saved successfully**.



Gambar 2.14. Keterangan berhasil mengubah status

- 2) Setelah menjadi **Teacher/Educator**, barulah dapat dilakukan pembuatan **Class**, dengan cara:
 - Kembali ke *dashboard* akun, kemudian klik **Classes**.



Gambar 2.15. Fitur Classes

- Klik **Create new class**.



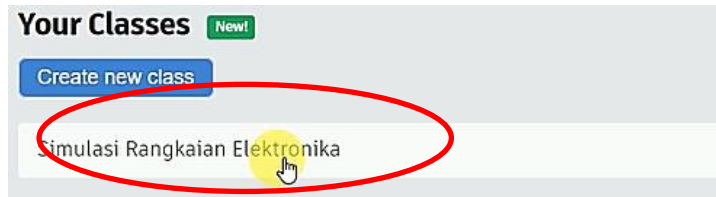
Gambar 2.16. Create new classes

- Pada **New class**, isi **Classroom name**, **Grades**, dan **Subject**, kemudian klik **Create class**. Berikut adalah contoh class.

A screenshot of the 'New class' dialog box in Tinkercad. The dialog has a title bar 'New class' with a close button. It contains three input fields: 'Classroom name' with the text 'Simulasi Rangkaian Elektronika' and a green checkmark icon; 'Grades' with a dropdown menu showing 'College'; and 'Subject' with a dropdown menu showing 'Electronics'. At the bottom right, there are two buttons: 'Cancel' and 'Create class'. A red circle highlights the 'Create class' button, which has a yellow mouse cursor icon pointing at it.

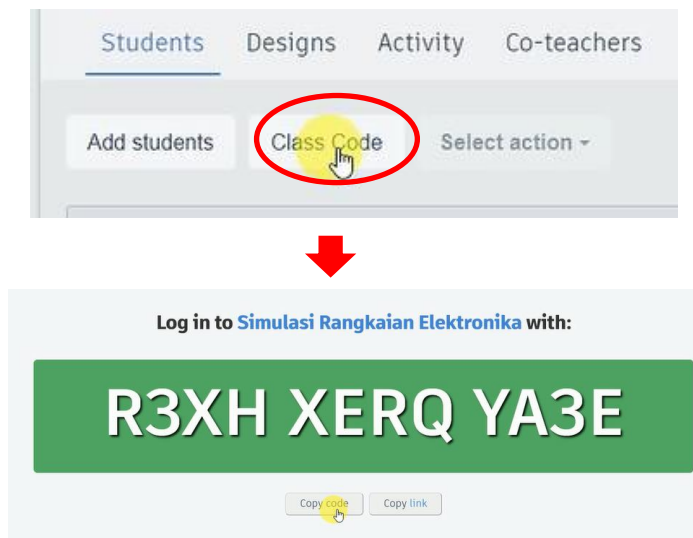
Gambar 2.17. Contoh class

- Jika sudah membuat class maka akan muncul tampilan seperti ini.



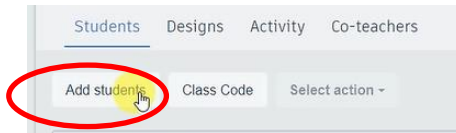
Gambar 2.18. Contoh class yang sudah dibuat

- Setelah class selesai dibuat, maka akan muncul class code yang dapat digunakan oleh mahasiswa/member untuk mengakses Tinkercad. Untuk melihat class code, klik nama class (Gambar 2.18), kemudian klik **class code**, maka akan muncul kode unik yang dapat di-copy.



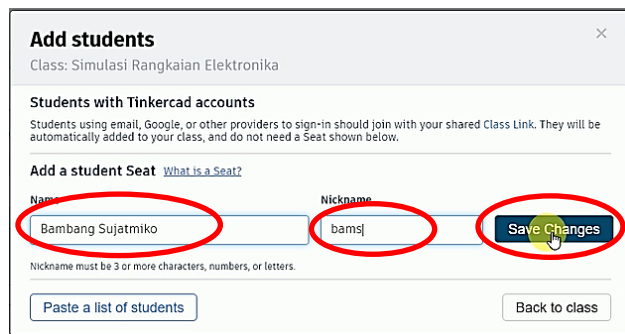
Gambar 2.19. Contoh class code

- 3) Selain menggunakan class code, mahasiswa/member suatu class harus memiliki **nickname** untuk dapat bergabung di suatu kelas. Nickname dibuat oleh Teacher/Educator dengan cara:
- Klik nama class seperti Gambar 2.18, kemudian pilih **Add students**.



Gambar 2.20. Add students

- Pada kotak **Add students**, masukkan nama dan nickname pada Kolom **Name** dan **Nickname**. Kemudian klik **Save Changes**. Untuk diketahui jika terdapat banyak nama yang ingin dimasukkan, maka tidak perlu diketik satu per satu melainkan bisa di-copy dari file lain kemudian di-paste ke kolom Name. Untuk nickname biasanya terisi secara otomatis, namun Teacher/Educator dapat mengubah sesuai keinginan seperti Gambar 2.21 di bawah.

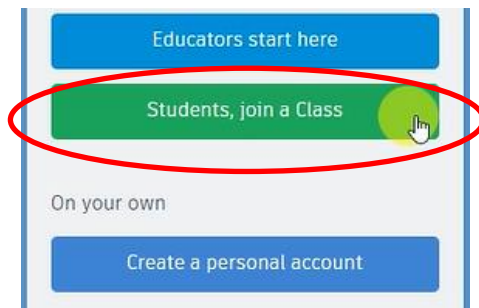
A screenshot of a form titled 'Add students' with a close button (X) in the top right. Below the title is the text 'Class: Simulasi Rangkaian Elektronika'. A section titled 'Students with Tinkercad accounts' contains a paragraph: 'Students using email, Google, or other providers to sign-in should join with your shared Class Link. They will be automatically added to your class, and do not need a Seat shown below.' Below this is a link 'What is a Seat?'. The form has two input fields: 'Name' with the value 'Bambang Sujatmiko' and 'Nickname' with the value 'bamsj'. Both fields are circled in red. To the right of the 'Nickname' field is a 'Save Changes' button, also circled in red with a yellow mouse cursor icon. At the bottom, there are two buttons: 'Paste a list of students' and 'Back to class'.

Gambar 2.21. Contoh pengisian Name dan Nickname

c. Join class

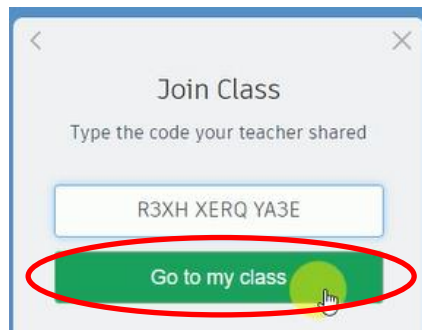
Setelah Teacher/Educator memberikan class code dan nick name, maka anggota dari suatu class dapat menggunakannya untuk join langsung ke class bahkan tanpa harus melakukan registrasi sebelumnya. Adapun cara join di class adalah sebagai berikut:

- 1) Masuk ke link <https://tinkercad.com/> kemudian pada menu awal Tinkercad klik **Join Now** (seperti Gambar 2.4). Selanjutnya pilih **Students, join a Class**.



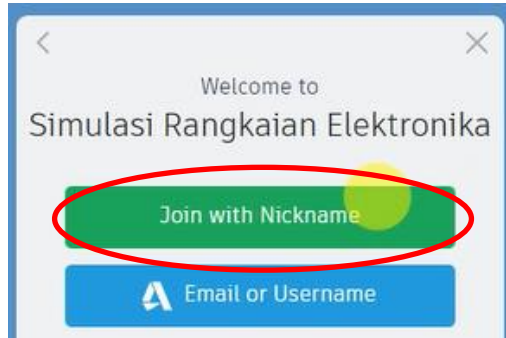
Gambar 2.22. Join class di Tinkercad

- 2) Masukkan class code. Contoh gunakan class code sesuai Gambar 2.19 kemudian klik Go to my class.



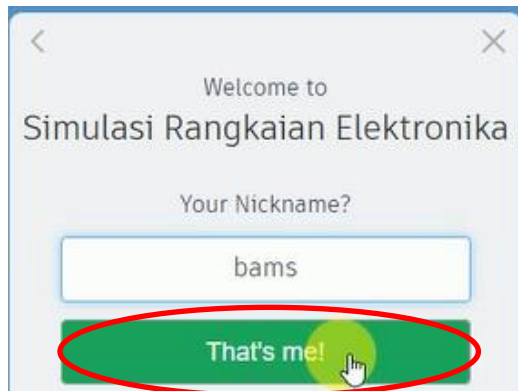
Gambar 2.23. Memasukkan class code

- 3) Setelah memasukkan class code akan muncul nama class. Selanjutnya pilih **Join with Nickname**.



Gambar 2.24. Muncul nama class

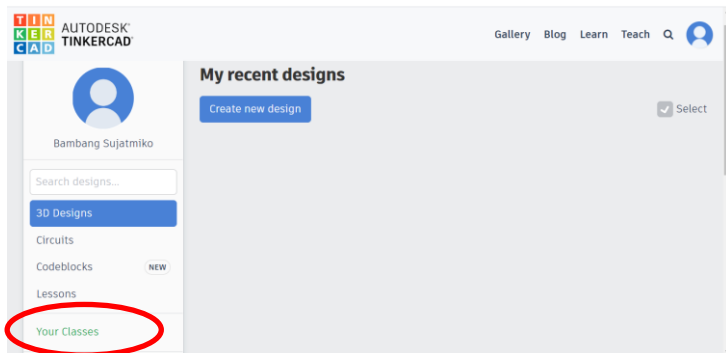
- 4) Masukkan nick name ke kolom **Your Nickname** (contoh gunakan nick name sesuai Gambar 2.21), kemudian klik **That's me**.



Gambar 2.25. Masukkan nick name

- 5) Jika sudah maka anggota class sudah dapat menggunakan Tinkercad. Hasil pengerjaan anggota

class dapat dimonitoring melalui akun Teacher/Educator yang membuat class. Gambar 2.26 memperlihatkan tampilan Tinkercad yang diakses melalui join class. Jika diperhatikan tidak ada yang berbeda dengan tampilan yang dimiliki oleh Teacher/Educator selain ketiadaan menu **Classes**. Anggota class dapat melihat nama class dan Teacher/Educator pada bagian **Your Classes**.

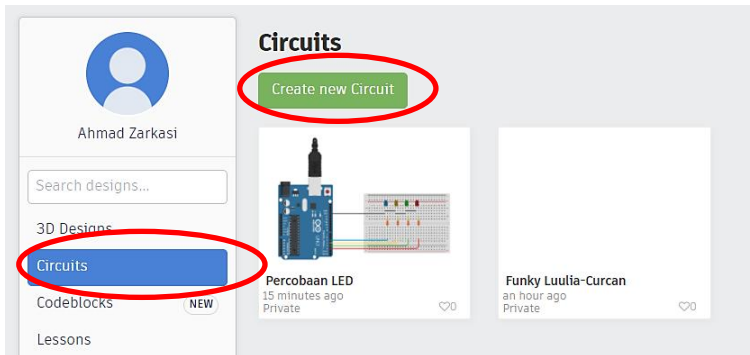


Gambar 2.26. Tampilan Tinkercad melalui join class

2.3 PERCOBAAN ARDUINO DI TINKERCAD

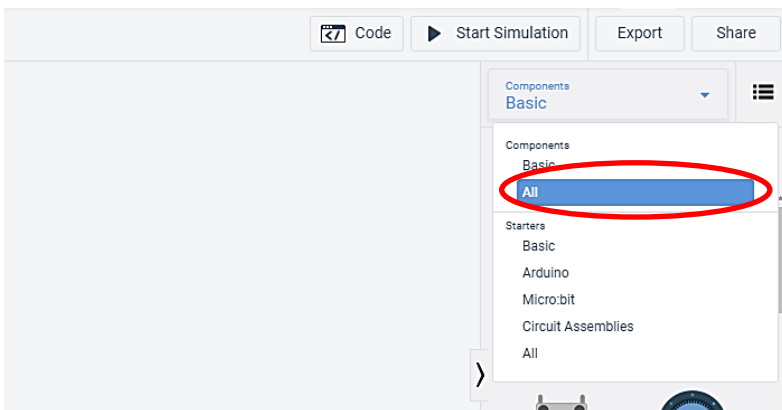
Pada bagian ini akan dijelaskan mengenai tahapan pembuatan simulasi berbasis Mikrokontroler Arduino melalui Tinkercad. Percobaan yang akan dilakukan berupa percobaan sederhana menggunakan 4 buah LED dengan tahapan sebagai berikut:

- a. Masuk ke akun Tinkercad yang sudah dibuat atau melalui join class menggunakan class code dan nick name. Selanjutnya pilih menu **Circuits** dan klik **Create new Circuit**.



Gambar 2.27. Create new Circuit

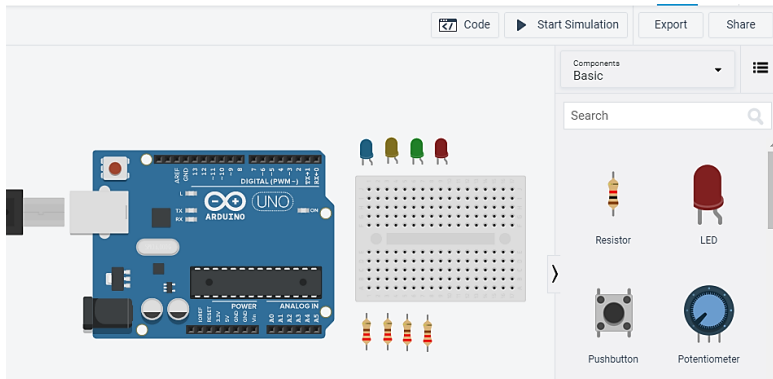
- b. Klik bagian **Components** kemudian pilih **All**. Hal ini bertujuan untuk dapat melihat semua pilihan komponen yang tersedia.



Gambar 2.28. Components

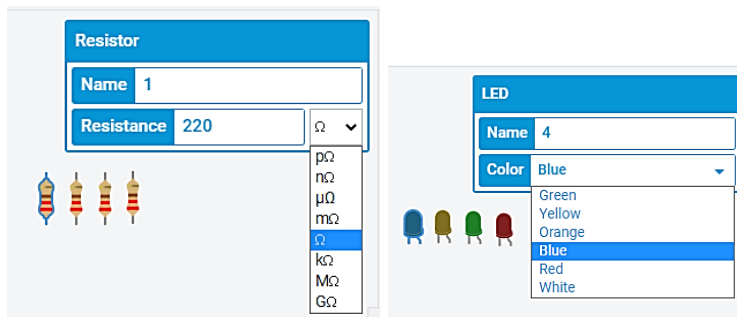
- c. Siapkan komponen yang diperlukan cara drag and drop. Adapun komponen yang diperlukan berupa:
 - 1) 1 unit *board* Arduino R3
 - 2) 1 buah bread *board*

- 3) 4 buah LED
- 4) 4 buah resistor 220 Ohm



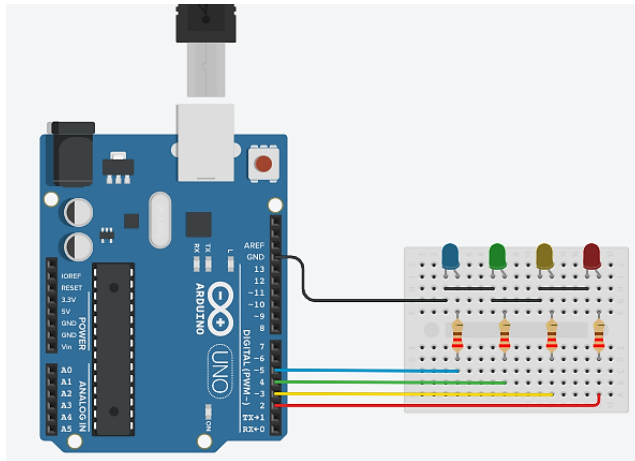
Gambar 2.29. Komponen yang diperlukan untuk simulasi

- d. Untuk mengubah nilai resistor maupun warna LED dapat dilakukan melalui properties yang tersedia pada masing-masing komponen.



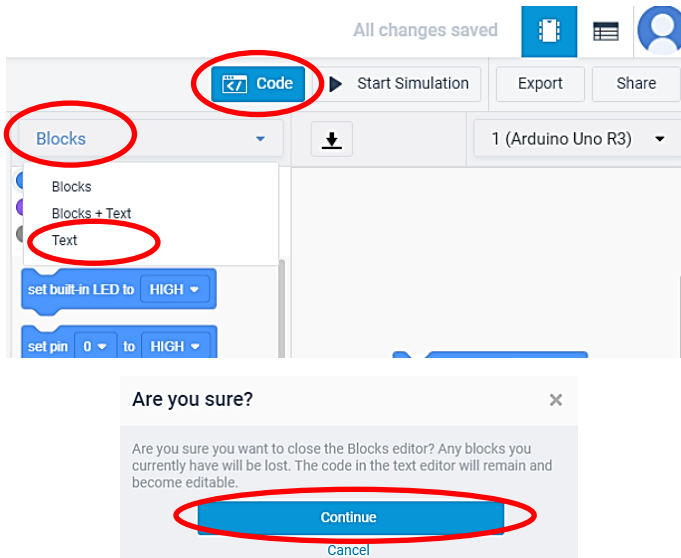
Gambar 2.30. Mengubah nilai resistor dan warna LED

- e. Susun rangkaian berdasarkan Gambar 2.31 berikut.



Gambar 2.31. Rangkaian simulasi

- f. Klik **Code** > **Blocks** > **Text**, lalu klik **Continue**.



Gambar 2.32. Memunculkan text editor

g. Pada **Text** ketikkan *sketch* berikut:

```
int LedB = 5; // Led Biru
int LedH = 4; // Led Hijau
int LedK = 3; // Led Kuning
int LedM = 2; // Led Merah

void setup()
{
  pinMode (LedB, OUTPUT);
  pinMode (LedH, OUTPUT);
  pinMode (LedK, OUTPUT);
  pinMode (LedM, OUTPUT);
}

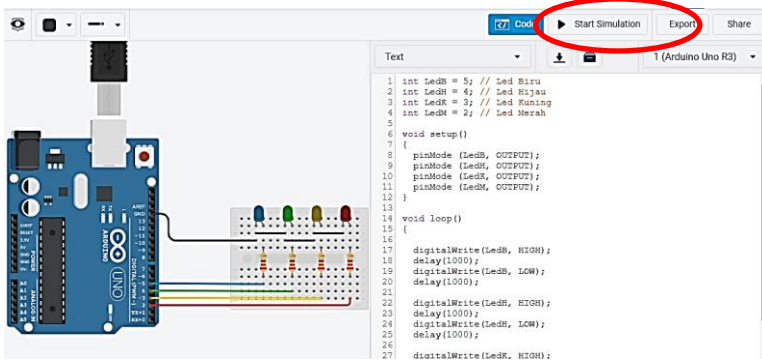
void loop()
{
  digitalWrite(LedB, HIGH);
  delay(1000);
  digitalWrite(LedB, LOW);
  delay(1000);

  digitalWrite(LedH, HIGH);
  delay(1000);
  digitalWrite(LedH, LOW);
  delay(1000);

  digitalWrite(LedK, HIGH);
  delay(1000);
  digitalWrite(LedK, LOW);
  delay(1000);

  digitalWrite(LedM, HIGH);
  delay(1000);
  digitalWrite(LedM, LOW);
  delay(1000);
}
```

h. Program siap dijalankan melalui **Start Simulation**.



Gambar 2.33. Start Simulation

BAB III

SYNTAX DASAR ARDUINO

Bahasa pemrograman yang digunakan untuk membuat *sketch* (program) pada Arduino adalah bahasa C++ namun dengan versi yang jauh lebih sederhana dan disesuaikan dengan kinerja hardware. Berikut beberapa instruksi dan *syntax* dasar dalam memprogram Arduino.

3.1 STRUCTURE

Struktur dasar bahasa pemrograman Arduino cukup sederhana dan berjalan setidaknya dengan memuat dua bagian utama yaitu **setup** dan **loop**.

```
void setup()  
{  
    statements; //Pernyataan  
}  
  
void loop()  
{  
    statements; //Pernyataan  
}
```

Blok **setup()** berisi persiapan program yang dieksekusi hanya sekali, sedangkan **loop()** berisi perintah yang dieksekusi secara berulang-ulang. Kedua bagian ini diperlukan agar program dapat berjalan. Fungsi **setup()** biasanya berisi deklarasi variabel, pengaturan **pinMode**, dan inisialisasi komunikasi serial. Sedangkan, fungsi **loop()** berisi kode yang dieksekusi secara terus menerus dan merupakan inti dari semua program Arduino karena sebagian besar program dieksekusi melalui fungsi ini.

a. Setup()

Fungsi **setup()** dipanggil sekali ketika program dijalankan. Biasanya **setup()** digunakan untuk mengatur mode dan komunikasi serial. Fungsi ini harus diikuti meskipun tidak mengandung pernyataan/perintah apapun.

```
void setup()
{
    pinMode (pin, OUTPUT); // Mengatur pin
                           // sebagai output
}
```

b. Loop()

Setelah memanggil fungsi **setup()**, fungsi **loop()** melakukan eksekusi secara terus-menerus dan dimulai dari program yang paling atas ke program yang paling bawah.

```
void loop()
{
    digitalWrite (pin, HIGH); // Memberi tegangan
                              // 5V ke pin
    delay (1000);             // Berhenti 1 detik

    digitalWrite (pin, LOW); // Memberi tegangan
                              // 0V pin
    delay (1000);             // Berhenti 1 detik
}
```

c. Functions (Fungsi)

Fungsi adalah kumpulan baris program yang dibuat untuk mengerjakan tugas tertentu. Pada dasarnya, fungsi dibuat untuk mengurangi penulisan kode program yang berulang. Berikut adalah struktur dari sebuah fungsi:

```
type function_name (parameters)
{
    body_of_the_function;
}
```

- **type**: Berupa tipe data seperti **int**, **float**, **bool**, dan lain-lain
- **function_name**: nama dari sebuah fungsi yang tidak boleh sama dengan *syntax* yang tersedia di Arduino IDE
- **parameter**: untuk memberikan nilai pada sebuah fungsi
- **function body**: merupakan tubuh dari fungsi tempat menulis baris-baris perintah yang ingin digunakan

Berikut contoh sebuah fungsi untuk menjumlahkan dua buah variabel. Fungsi **jumlahkan** menerima dua buah nilai berupa **num1** dan **num2** melalui **parameter**.

```
int jumlahkan (int num1, int num2){
{
    int hasil = num1 + num2;
    return hasil;
}
```

Agar dapat digunakan, suatu fungsi harus dipanggil. Misal:

```
jumlahkan (23, 91);
```

Untuk lebih memahami penggunaan fungsi, perhatikan contoh lain berikut ini.

```
int nilaiDelay()
{
```

```

int v;           // Membuat variabel 'v'
v = analogRead (pot); // Membaca nilai
                        // potensiometer
v /= 4;         // Konversi 0-1023 menjadi 0-255
return;         // Mengembalikan nilai ke program
}

```

Fungsi di atas merupakan fungsi konversi angka 0-1023 menjadi 0-255 yang digunakan untuk mengatur delay dalam program dengan membaca nilai potensiometer. Pertama digunakan sebuah tipe data integer dengan nama fungsi **nilaiDelay**. Sebuah variabel **v** digunakan untuk membaca nilai analog dari potensiometer. Nilai potensiometer yang berkisar antara 0-1023 kemudian dibagi 4 untuk memperoleh nilai akhir antara 0-255. Terakhir, nilai tersebut dikembalikan ke program utama.

d. Curly Brackets (kurung kurawal) { }

Kurung kurawal mendefinisikan awal dan akhir blok suatu fungsi atau pernyataan seperti fungsi **void loop()**, **for**, **if**, dan lain-lain. Namun, pada kenyataannya, suatu fungsi bisa saja tidak mengandung pernyataan sama sekali. Walaupun demikian, fungsi yang tidak mengandung pernyataan sekalipun harus menggunakan kurung kurawal.

```

type function ()
{
    statements;
}

```

Kurung kurawal buka '{' harus diikuti oleh kurung kurawal tutup '}', jika tidak maka akan menyebabkan terjadinya error pada saat melakukan *compile*.

e. Semicolon (Titik Koma) ;

Tanda titik koma harus digunakan untuk mengakhiri pernyataan dan memisahkan elemen program. Titik koma juga digunakan untuk memisahkan elemen dalam for loop.

```
int x = 13; // Mendeklarasikan variabel x
           // sebagai integer 13
```

Lupa dalam mengakhiri baris dengan titik koma akan menyebabkan kesalahan pada saat melakukan *compile*. Biasanya kesalahan semacam ini dapat dideteksi melalui keterangan yang muncul pada konsol *text editor* di Arduino IDE.

f. Block Comments (Komentar Blok) /*...*/

Komentar blok disebut juga komentar *multi line* merupakan area teks yang diabaikan oleh program. Komentar blok dipergunakan untuk mendeskripsikan suatu program sehingga memudahkan orang lain dalam memahami sebuah program. Penggunaan komentar blok ini memungkinkan penulisan banyak komentar (terdiri dari banyak baris).

```
/*
Ini merupakan contoh block comments yang tidak
dieksekusi namun sangat bermanfaat untuk
memahami sebuah program
*/
```

g. Line Comments (Komentar Baris) //

Komentar baris dimulai dengan tanda *'//'* dan diakhiri dengan kode berikutnya. Seperti komentar blok, komentar baris juga diabaikan oleh program dan tidak

memakan ruang memori. Komentar baris hanya digunakan untuk membuat keterangan yang hanya terdiri dari satu baris. Komentar baris biasanya dimanfaatkan untuk memberi keterangan suatu baris program agar mudah diingat.

```
// Ini adalah contoh komentar dalam satu baris
```

3.2 VARIABLES (VARIABEL)

Variabel adalah kode program yang digunakan untuk menyimpan suatu nilai pada sebuah nama. Seperti namanya, variabel adalah angka yang nilainya dapat diubah. Untuk menggunakannya, sebuah variabel perlu dideklarasikan dan secara opsional ditetapkan ke nilai yang perlu disimpan. Kode berikut mendeklarasikan variabel yang disebut **inputVariable** dan memberikan nilai yang diperoleh pada pin analog 2.

```
int inputVariable = 0;    // Mendeklarasikan variabel  
                        // dan memberikan nilai 0  
inputVariable = analogRead(2); // Mengatur var ke nilai  
                        // pin Analog 2
```

inputVariable merupakan nama variabel. Baris pertama menyatakan bahwa variabel tersebut akan berisi **int** (integer). Baris kedua menetapkan variabel ke nilai pada pin analog 2. Hal ini membuat nilai pada pin 2 dapat diakses di tempat lain dalam kode program.

Setelah suatu variabel ditetapkan, dapat dilakukan uji nilai untuk melihat apakah variabel tersebut memenuhi kondisi tertentu atau tidak. Di samping itu, nilai yang ada di dalam variabel juga dapat digunakan secara langsung. Berikut

adalah contoh kode program untuk mengatur delay berdasarkan nilai sebuah variabel.

```
if (inputVariable < 100) // Menguji variabel jika kurang
                        // dari 100
{
    inputVariable = 100; // Jika benar maka berikan nilai 100
}
delay(inputVariable); // Menggunakan variabel sebagai
                      // delay
```

Jika nilai **inputVariable** kurang dari 100 maka akan diatur agar memiliki nilai 100. Jika **inputVariable** sudah memiliki nilai 100, maka nilai tersebut akan digunakan untuk mengatur waktu delay.

a. Variabel Declaration (Deklarasi Variabel)

Semua variabel harus dideklarasikan sebelum dapat digunakan. Mendeklarasikan variabel berarti mendefinisikan tipe nilainya seperti dalam int, long, float, byte, dan lain-lain. Deklarasi variabel juga dilakukan untuk menetapkan nama tertentu dan secara opsional menetapkan nilai awalnya. Pada program, deklarasi variabel hanya perlu dilakukan sekali saja. Contoh berikut menyatakan bahwa **inputVariable** menggunakan tipe data int (integer atau bilangan bulat) yang nilainya sama dengan nol. Hal ini dinamakan *assignment* sederhana.

```
int inputVariable = 0;
```

b. Variable Scope (Cakupan Variabel)

Variabel dapat dideklarasikan pada tiga lokasi yang berbeda. Pertama, di bagian header atau di awal program sebelum **void setup()**, variabel ini disebut *global variable*.

Kedua, di dalam fungsi atau blok yang disebut *local variable*. Ketiga, di dalam definisi parameter fungsi yang disebut dengan *formal parameters*. Contoh berikut menunjukkan cara mendeklarasikan beberapa jenis variabel yang berbeda dan menunjukkan visibilitas setiap variabel.

```
int value; // variabel 'value' berlaku pada semua
           // fungsi
void setup()
{
    // Dapat dikosongkan
}
void loop()
{
    for (int i=0; i<20;) // 'i' hanya berlaku di
                        // dalam for loop
    {
        i++;
    }
    float f; // 'f' hanya berlaku dalam loop
}
```

3.3 DATA TYPES (TIPE DATA)

Tipe data adalah klasifikasi jenis data yang diperuntukkan agar *compiler* dapat mengetahui bagaimana sebuah data akan digunakan.

a. Byte

Byte menyimpan nilai numerik sebesar 8-bit tanpa titik desimal dengan jangkauan 0-255.

```
byte variabel1 = 180; // Deklarasi 'variabel1'
                     // dengan tipe byte
```

b. Integers (int)

Integer atau bilangan bulat adalah tipe data utama untuk penyimpanan angka tanpa titik desimal dengan ukuran 16-bit. Nilai tipe data ini berkisar antara -32.768 hingga 32.767.

```
int variabel2 = 10; // Deklarasi 'variabel2'
                // dengan tipe integer
```

c. Long

Long merupakan tipe data integer dengan ukuran yang diperluas menjadi 32-bit. Nilai long berkisar antara -2.147.483.648 sampai 2.147.483.647.

```
long variabel3 = 900000; // Deklarasi 'variabel3'
                        // dengan tipe long
```

d. Float

Tipe data ini digunakan untuk angka-angka desimal. Tipe data float memiliki resolusi yang lebih tinggi dari pada integer yaitu berukuran 32-bit dengan kisaran nilai -3,40282235E+38 sampai 3,40282235E+38

```
float variabel4 = 3.14; // Deklarasi 'variabel4'
                       // dengan tipe float
```

e. Array

Array adalah kumpulan nilai yang diakses dengan nomor indeks. Nilai apapun dalam array dapat dipanggil dengan memanggil nama array dan indeks nilainya. Indeks pada array dimulai dari 0. Array perlu dideklarasikan dan diberi nilai secara opsional sebelum digunakan.

```
int arrayKu [ ] = {nilai0, nilai1, nilai2 . . .}
```

Demikian juga dimungkinkan untuk mendeklarasikan array dengan mendeklarasikan tipe dan ukuran array serta menetapkan posisi indeks.

```
int arrayKu[5]; // Deklarasi array bertipe integer
                // pada posisi ke-6

arrayKu[3]=10; // Memberikan indeks 4 ke nilai 10
```

Array sering digunakan dalam for loop dimana penghitung kenaikan (increment counter) juga digunakan sebagai posisi indeks setiap nilai array. Contoh berikut menggunakan array untuk mengedipkan 4 buah LED secara bergantian.

```
int led[4]={5, 4, 3, 2}; // Array led memiliki
                        // indeks 0, 1, 2, dan 3

void setup(){
    for (int i=0; i<=3; i++){ // Mengatur indeks led
                                // yang akan dijadikan
        pinMode(led[i], OUTPUT); // sebagai output
    }
}

void loop(){
    for (int i=0; i<=3; i++){ // Mengedipkan led[0]
                                // sampai led[3]
        digitalWrite(led[i], HIGH);
        delay(1000);
        digitalWrite(led[i], LOW);
        delay(1000); // delay selama 1 detik
    }
}
```

Selain ke-5 tipe data di atas, masih banyak lagi tipe data lain seperti yang diperlihatkan Tabel 3.1.

Tabel 3.1. Jenis Tipe Data

Tipe data	Ukuran	Rentang Nilai
boolean	8 bit (1 byte)	1 atau 0 (True atau False)
byte	8 bit	0 sampai 255
char	8 bit	-128 sampai 127
unsigned char	8 bit	0 sampai 255
int	16 bit	-32.768 sampai 32.767
unsigned int	16 bit	0 sampai 65.535
word	16 bit	0 sampai 65.535
long	32 bit	-2.147.483.648 sampai 2.147.483.647
float	32 bit	-3,4028235E+38 sampai 3,4028235E+38
double	64 bit	-3,4028235E+38 sampai 3,4028235E+38
string	1 byte + x	Array dari char
array	8 bit + x	Kumpulan variabel

3.4 OPERATOR

a. Arithmetic Operators (Operator Aritmatika)

Tabel 3.2 memperlihatkan beberapa jenis operator aritmatika beserta fungsinya.

Tabel 3.2. Operator Aritmatika

Operator	Nama Operator	Fungsi
+	Addition (penjumlahan)	Melakukan penjumlahan
−	Subtraction	Melakukan pengurangan
*	Multiplication	Melakukan perkalian
/	Division	Melakukan pembagian
%	Modulo	Mencari sisa hasil bagi
=	Assignment operator	Menyimpan nilai yang ada di sebelah kanan tanda sama dengan dalam variabel yang ada di sebelah kiri tanda sama dengan

Berikut contoh penulisan program menggunakan operator aritmetika.

```
int x = 9; //deklarasi variabel 'x' bertipe integer
int y = 4; //deklarasi variabel 'y' bertipe integer
int z;

z = x + y; // menghasilkan 13
z = x - y; // menghasilkan 5
z = x * y; // menghasilkan 36
z = x / y; // menghasilkan 2
z = x % y; // menghasilkan 1
```

Ingat bahwa operasi dilakukan menggunakan tipe data operan. Jadi untuk x/y atau $9/4$ menghasilkan 2 dan bukan

2.25 karena 9 dan 4 adalah integer dan tidak dapat menggunakan titik desimal.

b. Comparison Operator (Operator Pembandingan)

Biasanya operator pembandingan digunakan pada pernyataan `if` untuk membandingkan suatu variabel/konstanta terhadap variabel/konstanta yang lain. Operator pembandingan sering digunakan untuk menguji suatu kondisi apakah bernilai **true** (**benar**) atau **false** (**salah**).

Tabel 3.3. Operator Pembandingan

Operator	Nama Operator	Fungsi
<code>==</code>	Equal	Mengecek apakah nilai dua operan sama atau tidak, jika sama maka hasilnya True
<code>!=</code>	Not equal	Mengecek apakah nilai dua operan sama atau tidak, jika tidak sama maka hasilnya true
<code><</code>	Less than	Mengecek apakah nilai operan di sebelah kiri bernilai kurang dari operan sebelah kanan, jika iya maka bernilai true
<code>></code>	Greater than	Mengecek apakah nilai operan di sebelah kiri bernilai lebih besar dari operan sebelah kanan, jika iya maka bernilai true

Operator	Nama Operator	Fungsi
<=	Less than or equal	Mengecek apakah nilai operan di sebelah kiri bernilai kurang dari atau sama dengan operan sebelah kanan, jika iya maka bernilai true
>=	Greater than or equal	Mengecek apakah nilai operan di sebelah kiri bernilai lebih besar dari atau sama dengan operan sebelah kanan, jika iya maka bernilai true

Berikut contoh penulisan program menggunakan operator pembandingan.

```
int a = 9;          // deklarasi variabel a
int b = 4;          // deklarasi variabel b
bool c; // deklarasi variabel c dengan tipe Boolean

if(a == b)          // Statement if ini menghasilkan
                    // false (0)
    c = true;
else
    c = false;

if(a != b)          // Statement if ini menghasilkan
                    // true (1)
    c = true;
else
    c = false;

if(a < b)           // Statement if ini menghasilkan
                    // false (0)
    c = true;
else
    c = false;
```

```
    if(a > b) // Statement if ini menghasilkan true
(1)      c = true;
    else
      c = false;

    if(a <= b)//Statement if ini menghasilkan false
(0)      c = true;
    else
      c = false;

    if(a >= b)// Statement if ini menghasilkan true
(1)      c = true;
    else
      c = false;
```

c. **Logical Operators (Operator Logika)**

Operator logika biasanya merupakan cara untuk membandingkan dua ekspresi sehingga nilai **true** atau **false** tergantung pada operatornya. Operator logika pada Arduino terdiri dari logika **AND**, **OR**, dan **NOT**.

Tabel 3.4. Operator Logika

Operator	Nama Operator	Fungsi
&&	AND	Bernilai true jika kedua ekspresi bernilai true
	OR	Bernilai true jika salah satu bernilai true
!	NOT	Bernilai true jika ekspresi bernilai false

Berikut contoh program menggunakan operator logika.

```
int a = 9,
b = 4
bool c ;
```

```

if((a > b)&& (b >= a))    // Menghasilkan false
    c = true;
else
    c = false;

if((a == b)|| (b < a))    // Menghasilkan true
    c = true;
else
    c = false;

if( !(a == b)&& (b < a)) // Menghasilkan true
    c = true;           // Gabungan dari NOT dan
                        // AND
else
    c = false;

```

d. Compound Operators (Operator Majemuk)

Operator majemuk menggabungkan operasi aritmetika dengan sebuah *variable assignment*. Operator ini biasanya digunakan pada for loop yang secara umum diperlihatkan pada Tabel 3.3.

Tabel 3.5. Operator Majemuk

Operator	Nama Operator	Fungsi
++	Increment	Menambahkan suatu nilai dengan 1
--	Decrement	Mengurangi suatu nilai dengan 1
+=	Compound addition	Merupakan bentuk ringkas dari operasi penjumlahan
-=	Compound subtraction	Merupakan bentuk ringkas dari operasi pengurangan

Operator	Nama Operator	Fungsi
<code>*=</code>	Compound multiplication	Cara untuk melakukan perkalian suatu variabel dengan konstanta atau variabel lain
<code>/=</code>	Compound division	Cara untuk melakukan pembagian suatu variabel dengan konstanta atau variabel lain
<code>% =</code>	Compound modulo	Cara mudah menghitung sisa pembagian
<code>& =</code>	Compound bitwise AND	Cara mudah menggunakan operator bitwise AND (bitwise <code>&</code> digunakan untuk mengkonversi nilai operan desimal menjadi bentuk biner dan melakukan operasi AND pada biner tersebut)
<code> =</code>	Compound bitwise OR	Cara mudah menggunakan operator bitwise OR (penggunaan bitwise <code> </code> sama dengan <code>&</code> akan tetapi <code> </code> digunakan pada operasi OR)

Berikut contoh program menggunakan operator majemuk:

```
/* Deklarasi variabel a, b, dan c dalam bentuk
integer*/
int a = 10;
int b = 20;
int c;

// Penggunaan operator majemuk
a++; // sama seperti a = a+1, bernilai 11
a--; // sama seperti a = a-1, bernilai 9
b += a; // sama seperti b = b+a, bernilai 30
b -= a; // sama seperti b = b-a, bernilai 10
b *= a; // sama seperti b = b*a, bernilai 300
b /= a; // sama seperti b = b/a, bernilai 2
b %= a; // sama seperti b = b%a, bernilai 0
a |= b; // bernilai 0
a &= b; // bernilai 0
```

3.5 CONSTANTS (KONSTANTA)

Bahasa pemrograman pada Arduino memiliki beberapa nilai yang telah didefinisikan yang disebut konstanta. Konstanta digunakan untuk membuat program lebih mudah dibaca. Konstanta diklasifikasikan ke dalam beberapa kelompok.

a. True/False

Merupakan konstanta Boolean yang mendefinisikan level logika. Konstanta **false** didefinisikan sebagai 0 (nol), sedangkan **true** didefinisikan sebagai 1 (satu) atau bisa apa saja kecuali 0. Jadi, dalam Boolean, nilai -1, 2, 200 dan lain-lain (kecuali 0) didefinisikan sebagai **true**. Berikut contoh penggunaannya.

```

if (b == true)    // Syarat agar perintah dijalankan
{
    lakukanSesuatu; // Berisi perintah yang ingin
                    // dijalankan
}

```

b. High/Low

Konstanta ini menentukan level pin Arduino sebagai **HIGH** atau **LOW** dan digunakan saat membaca atau menulis nilai pin digital. Konstanta **HIGH** didefinisikan sebagai level logika 1, ON, atau 5 Volt. Sedangkan, konstanta **LOW** adalah level logika 0, OFF, atau 0 Volt.

```

digitalWrite (13, HIGH); // Pin 13 Arduino diberi
                        // logika 1 yang berarti
                        // diberi tegangan 5V
digitalWrite (12, LOW);  // Pin 13 Arduino diberi
                        // logika 0 yang berarti
                        // diberi tegangan 0V

```

c. Input/Output

Pada Arduino terdapat pin I/O (Input/Output) yang dapat digunakan untuk mengakses atau menerima data, baik berupa data digital maupun data analog. Untuk menentukan apakah pin tersebut digunakan sebagai input atau output, perlu dilakukan pengaturan mode pin menggunakan fungsi **pinMode** dan konstanta **INPUT** atau **OUTPUT**.

```

pinMode (A0, INPUT); // Pin A0 sebagai input
pinMode (2, OUTPUT); // Pin 2 sebagai output

```

3.6 FLOW CONTROL

a. if

Pernyataan **if** digunakan untuk menguji apakah suatu kondisi telah terpenuhi atau tidak yang nantinya akan menentukan keputusan dalam mengeksekusi suatu perintah/ Pernyataan. Jika pernyataannya benar, maka perintah atau pernyataan apapun yang ada di dalam kurung akan dieksekusi. Jika salah, maka program akan melewati pernyataan tersebut. Pernyataan **if** biasanya menggunakan operator pembandingan. Berikut adalah *syntax* penulisan program menggunakan **if**.

```
if (sebuahVariabel ?? sebuahNilai)
{
    lakukanSesuatu; // Berisi pernyataan/perintah
}
```

Contoh di atas membandingkan **sebuahVariabel** dengan nilai lain yang dapat berupa variabel atau konstanta. Jika perbandingan atau kondisi dalam tanda kurung () benar, pernyataan di dalam tanda kurung kurawal { } akan dijalankan. Jika tidak, program akan melewatinya. Contoh penggunaan **if** dapat dilihat pada contoh penggunaan operator pembandingan dan operator logika.

b. if else

Penggunaan **if else** berfungsi untuk menjalankan salah satu perintah atau pernyataan berdasarkan kesesuaian hasil pengujian kondisi pada tanda kurung () **if**. Misalnya jika ingin menghidupkan dua buah LED secara bergantian yang dikendalikan menggunakan sebuah push button.

```

if (pushButton == HIGH) // Jika push button
                        // ditekan (HIGH)
{
    digitalWrite (ledHijau, HIGH); // Led hijau menyala
    digitalWrite (ledMerah, LOW);  // Led merah mati
}
else // Program di bawah dijalankan jika pushButton
    // == LOW
{
    digitalWrite (ledHijau, LOW); // Led hijau mati
    digitalWrite (ledMerah, HIGH); // Led merah menyala
}

```

Else juga dapat digunakan sebelum pengujian **if** yang lain (**else if**), sehingga beberapa pengujian yang eksklusif dapat dijalankan pada saat yang bersamaan. Selain itu, penggunaan **else if** memungkinkan percabangan pengujian dengan jumlah yang sangat banyak (secara teori tak terbatas). Ingat bahwa hanya satu perintah atau pernyataan yang akan dijalankan tergantung pada hasil pengujian kondisi.

```

if (inputPin < 500) // Pengujian kondisi 1
{
    doThingA; // Dieksekusi bila kondisi 1 terpenuhi
}
else if (inputPin >= 1000) // Pengujian kondisi 2
{
    doThingB; // Dieksekusi bila kondisi 1 tidak
              // terpenuhi dan kondisi 2 terpenuhi
}
else
{
    doThingC; // Dieksekusi bila kondisi 1 dan 2 tidak
              // terpenuhi
}

```

c. **for**

Pernyataan **for** digunakan untuk mengulang suatu pernyataan yang berada dalam kurung kurawal { }.

Perhitungan yang digunakan dapat berupa penambahan ataupun pengurangan suatu bilangan dengan batasan tertentu untuk mengakhiri proses looping. **For** banyak digunakan pada operasi yang berulang-ulang dengan kombinasi bilangan tertentu. Penggunaan **for** dapat mempersingkat *sketch* sehingga memperkecil ukuran suatu program. **For** memiliki 3 bagian yang dipisahkan oleh tanda titik koma (;). Berikut *syntax* penulisan program menggunakan **for**.

```
for (inisialisasi, kondisi, ekspresi/operasi)
{
    lakukanSesuatu; // Berisi pernyataan/perintah
}
```

Perhitungan variabel lokal dan *increment* dijalankan sekali dan paling awal. Setiap kali melalui loop dilakukan pengujian kondisi. Jika kondisi benar, pernyataan dan ekspresi akan dieksekusi dan pengujian kondisi kembali terjadi. Ketika kondisi salah maka loop akan berakhir. Contoh berikut memulai bilangan bulat **i** pada 0 dan melakukan pengujian pada nilai **i** apakah masih kurang dari 20. Jika benar, maka nilai **i** akan ditambah sebanyak 1 dan mengeksekusi sebuah perintah untuk menghidupkan dan mematikan LED sebanyak 20 kali (**i=0** sampai **i=19**).

```
/*Deklarasi 'i' bertipe integer. Kemudian menguji
nilai 'i' apakah i<20? Kemudian lakukan increment
senilai 1*/
for (int i=0; i<20; i++)
{
    digitalWrite(13, HIGH); // pin 13 ON
    delay(250);             // berhenti 1/4 detik
    digitalWrite(13, LOW);  // pin 13 OFF
    delay(250);             // berhenti 1/4 detik
}
```

d. **while**

Sebuah perintah pada **while** loop akan dieksekusi secara terus menerus dan tanpa henti sampai terdapat kondisi di dalam kurung yang bernilai salah. Berikut adalah *syntax* penulisan **while**.

```
while (kondisi ?? nilai)
{
    lakukanSesuatu; // Berisi pernyataan/perintah
}
```

Contoh berikut ini menguji apakah **suatuVariabel** memiliki nilai kurang dari 200. Jika benar maka akan dilakukan *increment* sejumlah 1 secara terus menerus sampai batas < 200.

```
while (suatuVariabel < 200) // Mengecek apakah nilai
{
    // 'variabelKu' kurang
    // dari 200
    suatuVariabel++; // increment yang dapat juga
    // ditulis:
    // suatuVariabel = suatuVariabel + 1
}
```

e. **do-while**

Pernyataan perulangan **do while** hampir sama dengan pernyataan **while**. Perbedaannya yaitu jika pada pernyataan **while**, kondisi diuji dahulu dan bila uji kondisi bernilai benar maka pernyataan yang ada dalam blok **while** akan dieksekusi. Sedangkan, pada **do while** kondisi menjadi terbalik yaitu pernyataan utama akan dieksekusi terlebih dahulu, setelah itu baru dilakukan uji kondisi. Jika kondisi benar maka pernyataan utama akan diulang, tetapi jika salah program akan keluar dari blok **do while**. Berikut *syntax* penulisan **do while**.

```
do
{
lakukanSesuatu;
} while (suatuVariabel ?? nilai);
```

Di bawah ini adalah contoh penggunaan **do while**. Pertama-tama program akan melakukan delay selama 50 milidetik menunggu hingga sensor stabil. Selanjutnya program akan membaca sensor dan menyimpannya dalam variabel **x**. Kemudian, variabel **x** akan diuji apakah bernilai lebih kecil dari 100 atau tidak. Jika nilai **x** lebih kecil dari 100, maka program akan melakukan pengulangan (kembali membaca sensor), tetapi jika nilai **x** lebih dari 100, maka program akan keluar dari blok **do while** untuk melakukan perintah yang lain.

```
do {
delay (50);
x = readSensors(); // memasukkan nilai sensor ke
                    // variabel 'x'
} while (x < 100); // looping terjadi saat 'x' kurang
                  // dari 100
```

3.7 DIGITAL I/O

a. **pinMode (pin, mode)**

Syntax **pinMode (pin, mode)** digunakan dalam **void setup()** untuk mengkonfigurasi pin tertentu agar berperan sebagai input atau output seperti contoh di bawah.

```
pinMode (13, OUTPUT); // Mengatur pin 13 sebagai
                      // output
pinMode (3, INPUT);  // Mengatur pin 3 sebagai
                      // input
```

Pin digital pada Arduino secara *default* berperan sebagai input, sehingga pada dasarnya tidak perlu dideklarasikan

secara eksplisit sebagai input dengan **pinMode**. Pin yang dikonfigurasi sebagai input dikatakan dalam keadaan impedansi tinggi.

b. digitalWrite (pin)

Syntax **digitalRead (pin)** digunakan untuk membaca nilai dari pin digital tertentu dengan hasil **HIGH** atau **LOW**. Pin dapat ditentukan sebagai variabel atau konstanta.

```
/* Mengatur variabel 'nilai' agar sama dengan pin input*/  
nilai = digitalRead (pin);
```

c. digitalWrite (pin, nilai)

Syntax **digitalWrite (pin, nilai)** berfungsi memberikan logika **HIGH** atau **LOW** yang berarti memberikan nilai 1 (5V) atau 0 (0V) pada pin Arduino. Berikut contoh kode program untuk membaca nilai pada push button yang terhubung ke input digital dan menyalakan LED yang terhubung ke output digital ketika push button ditekan.

```
int led = 13; // Menghubungkan LED ke pin 13  
int pb = 7;   // Menghubungkan push button ke pin 7  
int nilaiPb = 0; // Variabel untuk menyimpan nilai Pb  
  
void setup() {  
  pinMode(led, OUTPUT); // Mengatur led sebagai output  
  pinMode(pb, INPUT);   // Mengatur push button sebagai input  
}  
void loop() {  
  nilaiPb = digitalRead(pb); // Membaca nilai pb dan  
                             // menyimpannya ke 'nilaiPb'  
  digitalWrite(led, nilai); // Agar 'led' sesuai dengan  
                             // 'nilaiPb'  
}
```

3.8 ANALOG I/O

a. `analogRead (pin)`

Syntax **`analogRead (pin)`** digunakan untuk membaca nilai dari pin analog Arduino dengan resolusi ADC tertentu sesuai spesifikasi *board* Arduino yang digunakan. Sebagai contoh, Arduino Uno memiliki resolusi ADC sebesar 10 bit yang dapat diakses pada pin A0 sampai A5. Nilai integer yang dihasilkan berkisar antara 0 sampai 1023.

```
value = analogRead (pin); //Membaca nilai analog pada
                          // 'pin' dan menyimpannya
                          //dalam variabel 'value'
```

b. `analogWrite (pin, nilai)`

Syntax **`analogWrite (pin, nilai)`** berfungsi untuk menulis nilai *pseudo-analog* melalui pin PWM. Pada masing-masing *board* Arduino, jumlah pin yang memiliki fitur PWM berbeda-beda tergantung jenis *chip* yang digunakan. Contohnya pada Arduino yang menggunakan *chip* ATmega328, fungsi PWM dapat bekerja pada pin 3, 5, 6, 9, 10, dan 11. Sementara pada Arduino yang lebih lama dengan *chip* ATmega8 hanya mendukung fungsi PWM pada pin 9, 10, dan 11. Kebanyakan Arduino memiliki PWM dengan resolusi sebesar 8 bit yang artinya dapat mengeluarkan nilai dari 0 sampai 255.

```
analogWrite (pin, value); //Menulis nilai analog
                          //sebesar 'value' ke 'pin'
```

Contoh berikut membaca nilai analog dari potensiometer sebagai input analog, kemudian nilai tersebut diubah dengan cara membaginya dengan 4, dan terakhir mengeluarkan sinyal PWM pada pin PWM.

```

int led = 10; // Menghubungkan led ke pin 10
int pot = A0; // Potensiometer terhubung dengan pin 0
int nilaiPot; // Membaca nilai potensio

void setup(){} // Tidak dilakukan setup
void loop()
{
  nilaiPot = analogRead(pot); // Membaca nilai
                                // potensiometer

  nilaiPot /= 4;                // Konversi 0-1023
                                // menjadi 0-255

  analogWrite(led, nilaiPot); // Output PWM ke led
                                // (berdasarkan
                                // nilai potensiometer)
}

```

3.9 TIME

a. delay (ms)

Syntax **delay (ms)** digunakan untuk memberikan jeda pada program untuk jumlah waktu yang ditentukan dalam satuan milidetik. Berikut contoh penulisan delay untuk jeda waktu selama 1 detik atau 1000 milidetik.

```

delay (1000); // Memberi jeda selama 1000 milidetik
              // atau 1 detik

```

b. millis ()

Syntax **millis ()** berguna untuk menjalankan waktu internal setiap milidetik pada Arduino secara independen. Ketika **millis** dibaca, maka **millis** akan terus menghitung waktu walaupun Arduino sedang menjalankan program yang lain. Berikut adalah contoh penulisan **millis**.

```

value = millis(); // Mengatur 'value' sama dengan
                  // nilai millis

```

3.10 MATH

a. **min (x, y)**

Syntax **min (x, y)** berguna untuk menghitung minimal dua angka dari tipe data apapun dan mengembalikan angka yang lebih kecil.

```
value = min(value, 100); // Mengatur 'value' menjadi
                          // lebih kecil dari 'value'
                          // atau 100 dan memastikan
                          // 'value' tidak akan di
                          // atas 100
```

b. **max (x, y)**

Syntax **max (x, y)** merupakan kebalikan dari **min (x,y)**. *Syntax* ini digunakan untuk menghitung maksimal dua angka dari tipe data apapun dan mengembalikan angka yang lebih besar

```
value = min(value, 100); // Mengatur 'value' menjadi
                          // lebih besar dari 'value'
                          // atau 100 dan memastikan
                          // 'value' tidak akan di
                          // bawah 100
```

3.11 SERIAL

a. **Serial.begin (rate)**

Pada Arduino terdapat fitur serial monitor dan serial plotter yang dapat dimanfaatkan untuk melihat hasil komunikasi serial berupa output suatu sensor, push button, dan lain-lain. *Syntax* **Serial.begin (rate)** digunakan untuk membuka port serial dan mengatur baud rate (kecepatan transmisi) untuk transmisi data serial. Baud rate tipikal untuk berkomunikasi dengan komputer adalah 9600 bps meskipun kecepatan lainnya masih tetap mendukung. Saat

menggunakan komunikasi serial, pin digital 0 (RX) dan 1 (TX) tidak dapat digunakan secara bersamaan. Berikut contoh pengaturan baudrate.

```
void setup()
{
    Serial.begin(9600); // Membuka port serial dengan &
                        // mengatur baudrate sebesar
                        // 9600 bps
}
```

b. **Serial.print (data) dan Serial.println (data)**

Syntax **Serial.print (data)** berfungsi mengirimkan data ke port serial dan menampilkannya pada serial monitor dalam satu baris saja. Jika argumen yang dimasukkan ke dalam perintah, maka data yang dikirim akan menyesuaikan dengan format tersebut.

```
Serial.print (91);           // Mencetak "91"
Serial.print ("Hello World!"); // Mencetak "Hello World"
Serial.print (variabel1)     // Mencetak nilai pada
                             // "variabel1"
Serial.print (91, BIN);      // Mencetak "01011011"
Serial.print (91, HEX);      // Mencetak "5B"
```

Syntax lain yang fungsinya hampir sama dengan **Serial.print (data)** adalah **Serial.println (data)**. *Syntax* ini digunakan juga untuk mengirimkan data yang selanjutnya ditampilkan di layar serial monitor. Jika dengan **Serial.print** data ditampilkan pada satu baris, maka dengan menggunakan **Serial.println** akan memunculkan baris baru.

```
Serial.println (variabel2) // Mencetak nilai pada
                          // "variabel2"
```


Perlu diingat bahwa sebelum menggunakan *syntax* **Serial.print (data)**, port serial harus dibuka dan kecepatan transmisi data harus diatur menggunakan *syntax* **Serial.begin (rate)**.

BAB IV

PERCOBAAN MIKROKONTROLER ARDUINO

Percobaan-percobaan yang dimuat pada buku ini melibatkan komponen-komponen sederhana dan sangat mudah didapatkan. Pada buku ini terdapat 8 buah percobaan yang dimulai dari percobaan paling sederhana menggunakan LED, sampai pada pembuatan *project* dengan menggunakan Arduino Uno sebagai *board* Mikrokontrolernya. Platform Tinkercad berperan menyediakan sarana simulasi yang interaktif karena memiliki *interface* yang sangat mudah digunakan dengan tampilan yang hampir mirip dengan rangkaian aslinya. Capaian akhir yang diinginkan setelah melakukan percobaan-percobaan ini adalah agar pengguna dapat merancang sebuah *project* menggunakan mikrokontroler yang dapat diaplikasikan dalam kehidupan secara nyata. Pembuatan *project* yang dimaksud tentu tidak terbatas pada percobaan-percobaan yang telah tersedia pada buku ini, namun lebih jauh lagi pengguna dapat melakukan banyak improvisasi dan pengembangan yang lebih *advance*.

4.1 PERCOBAAN 1 – LED

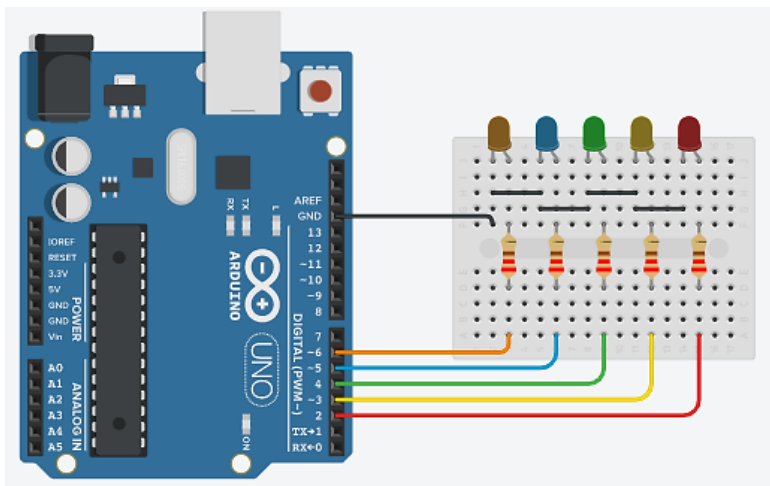
a. Tujuan Percobaan

- 1) Menggunakan output digital pada Arduino
- 2) Membuat program untuk mengendalikan LED
- 3) Mempersingkat penulisan *sketch* untuk menghemat memori

b. Alat dan Bahan

- 1) Arduino Uno
- 2) *Bread board*
- 3) 6 buah LED
- 4) 6 buah Resistor 220 Ω
- 5) Beberapa kabel penghubung

c. Rangkaian



Gambar 4.1. Rangkaian Percobaan 1

d. Program / Sketch

```
int led1 = 6; // LED orange
int led2 = 5; // LED biru
int led3 = 4; // LED hijau
int led4 = 3; // LED kuning
int led5 = 2; // LED merah

void setup()
{
  pinMode (led1, OUTPUT);
  pinMode (led2, OUTPUT);
  pinMode (led3, OUTPUT);
  pinMode (led4, OUTPUT);
  pinMode (led5, OUTPUT);
}

void loop()
{
  //led1 berkedip sebanyak 1 kali
  digitalWrite(led1, HIGH);
  delay(500);
  digitalWrite(led1, LOW);
  delay(500);

  //led2 berkedip sebanyak 2 kali
  digitalWrite(led2, HIGH);
  delay(500);
  digitalWrite(led2, LOW);
  delay(500);
  digitalWrite(led2, HIGH);
  delay(500);
  digitalWrite(led2, LOW);
  delay(500);

  //led3 berkedip sebanyak 3 kali
  digitalWrite(led3, HIGH);
  delay(500);
  digitalWrite(led3, LOW);
  delay(500);
  digitalWrite(led3, HIGH);
  delay(500);
  digitalWrite(led3, LOW);
  delay(500);
  digitalWrite(led3, HIGH);
  delay(500);
  digitalWrite(led3, LOW);
  delay(500);
}
```

```

//led4 berkedip sebanyak 4 kali
digitalWrite(led4, HIGH);
delay(500);
digitalWrite(led4, LOW);
delay(500);
digitalWrite(led4, HIGH);
delay(500);
digitalWrite(led4, LOW);
delay(500);
digitalWrite(led4, HIGH);
delay(500);
digitalWrite(led4, LOW);
delay(500);
digitalWrite(led4, HIGH);
delay(500);
digitalWrite(led4, LOW);
delay(500);

//led5 berkedip sebanyak 5 kali
digitalWrite(led5, HIGH);
delay(500);
digitalWrite(led5, LOW);
delay(500);
digitalWrite(led5, HIGH);
delay(500);
digitalWrite(led5, LOW);
delay(500);
digitalWrite(led5, HIGH);
delay(500);
digitalWrite(led5, LOW);
delay(500);
digitalWrite(led5, HIGH);
delay(500);
digitalWrite(led5, LOW);
delay(500);
digitalWrite(led5, HIGH);
delay(500);
digitalWrite(led5, LOW);
delay(500);
}

```

e. Instruksi dan Latihan

1) Instruksi

Buatlah rangkaian dan program sesuai dengan poin **c** dan **d**, kemudian jalankan program.

2) Latihan

Berdasarkan Gambar 4.1 buatlah program yang lebih ringkas namun dengan tujuan yang sama. Silakan terapkan penggunaan **fungsi**, **for**, **array**, dan lain-lain agar penggunaan memori menjadi lebih efisien.

4.2 PERCOBAAN 2 – PUSH BUTTON

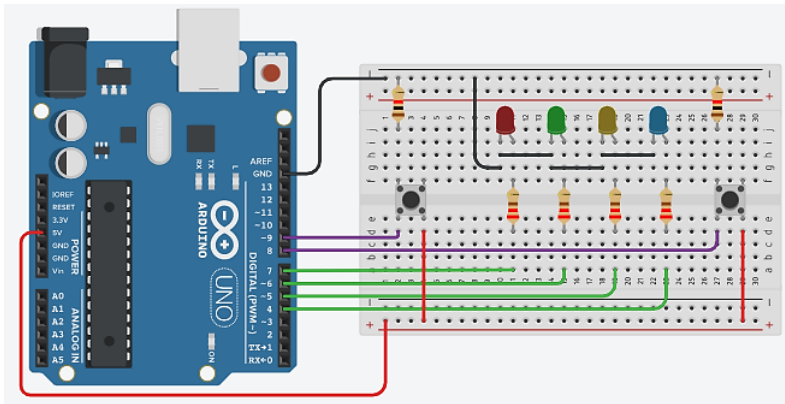
a. Tujuan Percobaan

- 1) Menggunakan input digital pada Arduino
- 2) Membuat program untuk membaca nilai digital dari push button melalui serial monitor
- 3) Menerapkan penggunaan **switch case** untuk mengendalikan LED

b. Alat dan Bahan

- 1) Arduino Uno
- 2) *Bread board*
- 3) 2 buah push button
- 4) 2 buah Resistor 1 k Ω
- 5) 4 buah Resistor 220 Ω
- 6) Beberapa kabel penghubung

c. Rangkaian



Gambar 4.2. Rangkaian Percobaan 2

d. Program / Sketch

```
const int pb1 = 8; // Push button 1
const int pb2 = 9; // Push button 2
int j;

void setup() {
  pinMode(pb1, INPUT);
  pinMode(pb2, INPUT);

  Serial.begin(9600);
}

void loop() {
  int nilaiPb1 = digitalRead(pb1);
  int nilaiPb2 = digitalRead(pb2);

  if(nilaiPb1==HIGH) {
    j++;
    delay(20);
  }

  if(nilaiPb2==HIGH) {
    j--;
    delay(20);
  }
  Serial.println(j);
}
```

e. Instruksi dan Latihan

1) Instruksi

- Buatlah rangkaian dan program sesuai dengan poin c dan d, kemudian jalankan program.
- Buka serial monitor, kemudian tekan push button secara bergantian. Amati nilai yang terbaca pada serial monitor.
- Ubah-ubah nilai j lalu amati nilai yang keluar di serial monitor.

2) Latihan

- Masih dengan rangkaian yang sama sesuai Gambar 4.2, kendalikan nyala LED dengan ketentuan:
 - LED 1 (merah) akan menyala ketika $j = 1$
 - LED 2 (hijau) akan menyala ketika $j = 2$
 - LED 3 (kuning) akan menyala ketika $j = 3$
 - LED 4 (biru) akan menyala ketika $j = 4$
 - Semua LED akan mati ketika $j < 1$ atau $j > 4$
 - Nilai j dikendalikan oleh kedua push button
- Gunakan perintah **switch case** untuk mengendalikan push button.

4.3 PERCOBAAN 3 – ADC dan PWM

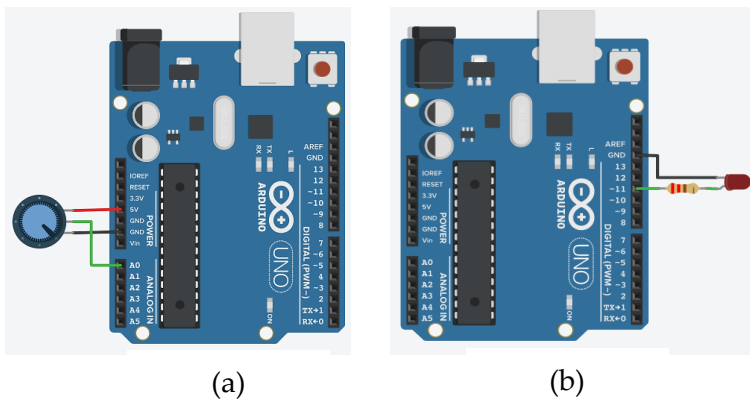
a. Tujuan Percobaan

- 1) Menerapkan penggunaan PWM dan ADC
- 2) Menggunakan channel analog dan PWM pada Arduino
- 3) Mengendalikan LED berdasarkan nilai voltase sebuah potensiometer

b. Alat dan Bahan

- 1) Arduino Uno
- 2) Potensiometer
- 3) Resistor 220 Ω
- 4) LED
- 5) Beberapa kabel penghubung

c. Rangkaian



Gambar 4.3. Rangkaian Percobaan 3 (a) ADC dan (b) PWM

d. Program / Sketch

1) Program ADC

```
const int pot = A0; // Channel analog
int nilaiPot;

float nilaiSerial; // Agar nilaiSerial berbentuk
                  // desimal

void setup() {
  Serial.begin(9600); // Memulai komunikasi serial
  pinMode (A0, INPUT);
}
void loop() {
  nilaiPot = analogRead(pot); // Membaca nilai
  potensio
  nilaiSerial = nilaiPot*(5.0/1023); // Konversi
  nilai
                                     // analog
                                     // ke digital (ADC)
  Serial.print("Nilai input = ");
  Serial.print(nilaiPot);
  Serial.print(" = ");
  Serial.print(nilaiSerial);
  Serial.println(" Volt");
  delay(200);
}
```

2) Program PWM

```
int led = 11; // pin PWM

void setup(){
  pinMode(led, OUTPUT);
}

void loop()
{
  for (int i = 0; i<255; i +=5) // nilai bertambah 5
                                // tiap 30 ms
  {
    analogWrite (led, i);
    delay(30);
  }
  for (int i = 255; i>=0; i -=5) // nilai berkurang 5
                                // tiap 30 ms
  {
```

```
analogWrite (led, i);  
    delay(30);  
}  
  
}
```

e. Instruksi dan Latihan

1) Instruksi

- Buatlah rangkaian serta program ADC dan PWM secara terpisah (dibuat masing-masing).
- Pada percobaan PWM, amati perubahan intensitas cahaya LED.
- Pada percobaan ADC, buka serial monitor dan putar potensiometer. Amati perubahan nilai yang terjadi.

2) Latihan

- Buatlah sebuah rangkaian baru dengan menggabungkan kedua rangkaian pada Gambar 4.3 menggunakan 1 *board* Arduino.
- Buatlah program untuk mengatur intensitas cahaya LED berdasarkan nilai potensiometer. Artinya, intensitas cahaya LED akan berubah ketika potensiometer diputar.

4.4 PERCOBAAN 4 – LCD

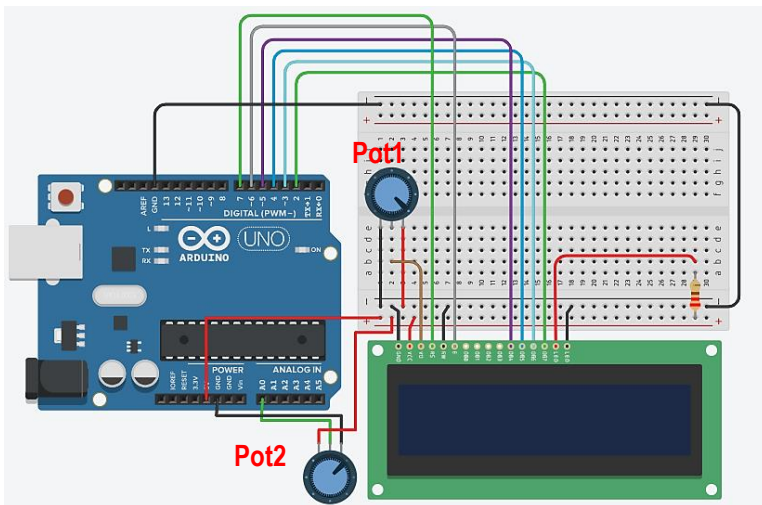
a. Tujuan Percobaan

- 1) Membuat rangkaian minimum untuk menggunakan LCD 16x2
- 2) Menerapkan penggunaan LCD sebagai display suatu input/output

b. Alat dan Bahan

- 1) Arduino Uno
- 2) LCD 16x2
- 3) 2 buah potensiometer
- 4) Breadboard
- 5) Beberapa kabel penghubung

c. Rangkaian



Gambar 4.4. Rangkaian Percobaan 4

d. Program / Sketch

```
#include<LiquidCrystal.h>           //Library LCD
Const int RS=7, EN=6, D4=5, D5=4, D6=3, D7=2;
LiquidCrystal lcd (RS,EN,D4,D5,D6,D7); //Inisialisasi
                                     //pin LCD

int pot = A0;

void setup(){
  lcd.clear();
  lcd.begin(16,2); // Memulai penggunaan LCD
  lcd.setCursor(0,0);
  lcd.print("Data");

  pinMode (pot, INPUT);
}

void loop(){
  int nilaiPot = analogRead(pot); // Membaca input
  lcd.setCursor(0,1);
  lcd.print(nilaiPot); // Menampilkan data dari
                       // potensiometer
}
```

e. Instruksi dan Latihan

1) Instruksi

- Buatlah rangkaian dan program seperti pada poin **c** dan **d**, lalu jalankan.
- Atur kontras LCD menggunakan **Pot1**.
- Putar **Pot2** untuk melihat perubahan data melalui LCD.

2) Latihan

- Modifikasi rangkaian pada Gambar 4.4 dengan menambahkan 3 buah LED.
- Konversi data input dari potensiometer menjadi satuan tegangan (0 - 5Volt)
- Kendalikan nyala LED dengan ketentuan:

- LED 1 menyala ketika $1 \text{ Volt} \leq \text{tegangan} < 2 \text{ Volt}$
- LED 2 menyala ketika $2 \text{ Volt} \leq \text{tegangan} < 3 \text{ Volt}$.
- LED 3 menyala ketika $3 \text{ Volt} \leq \text{tegangan} < 4 \text{ Volt}$.
- Semua LED mati jika $\text{tegangan} < 1 \text{ Volt}$ atau $\text{tegangan} \geq 4 \text{ Volt}$.

4.5 PERCOBAAN 5 – KEYPAD

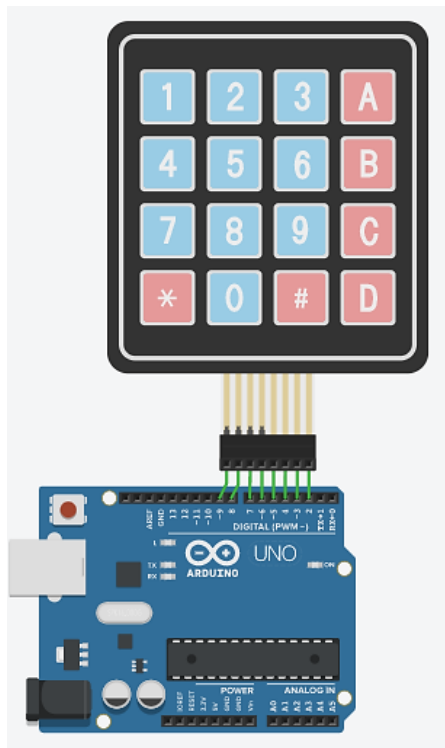
a. Tujuan Percobaan

Menerapkan penggunaan keypad untuk mengendalikan suatu output

b. Alat dan Bahan

- 1) Arduino Uno
- 2) Keypad 4x4
- 3) Beberapa kabel penghubung

c. Rangkaian



Gambar 4.5. Rangkaian Percobaan 5

d. Program / Sketch

```
#include <Keypad.h> // Memanggil library Keypad
const byte Baris = 4; // Jumlah baris
const byte Kolom = 4; // Jumlah kolom

//Memetakan tombol keypad
char petaTombol[Baris] [Kolom] =
{
  {'1','2','3','A'}, // Karakter-karakter yang
  {'4','5','6','B'}, // tersedia di keypad
  {'7','8','9','C'},
  {'*','0','#','D'},
};

//Koneksi Arduino dengan keypad
byte pinBaris[Baris] = {9, 8, 7, 6};
byte pinKolom[Kolom] = {5, 4, 3, 2};

//Definisi keypad
Keypad tombol =
Keypad(makeKeymap(petaTombol),pinBaris,pinKolom,
Baris,Kolom);

void setup() {
  Serial.begin(9600);
}

void loop() {
  // Variabel untuk menyimpan data saat tombol
  // ditekan
  char tombolDitekan = tombol.getKey();

  //Kondisi jika tombol ditekan
  if(tombolDitekan != NO_KEY) // Kondisi jika tombol
                              // ditekan

    Serial.print(tombolDitekan);
}
```

e. Instruksi dan Latihan

1) Instruksi

Buatlah rangkaian dan program sesuai poin c dan d, jalankan program kemudian amati serial monitor.

2) **Latihan**

Modifikasi rangkaian dan program yang ada, kemudian gunakan tombol-tombol pada keypad untuk mengendalikan output tertentu seperti LED, buzzer, tampilan LCD, servo, dan lain-lain (bebas).

d. Program / Sketch

```
#include <LiquidCrystal.h>
const int RS=3, EN=4, D4=9, D5=10, D6=11, D7=12;
LiquidCrystal lcd (RS, EN, D4, D5, D6, D7);

int TMP36 = A0; // Inisiasi input sensor

void setup(){
  lcd.clear();
  lcd.begin (16,2);
  lcd.setCursor (0,0);
  lcd.print ("Nilai Suhu");
}

void loop(){
  int baca = analogRead (TMP36); // Membaca nilai
  sensor
  float tegangan = baca * (5.0/1023); // Konversi
  nilai
  float suhu = (tegangan - 0.5)*100;

  lcd.setCursor(0,1);
  lcd.print(suhu);
  lcd.print("  Celcius");
}
```

e. Instruksi dan Latihan

1) Instruksi

- Buatlah rangkaian dan program sesuai poin c dan d, jalankan program.
- Klik sensor TMP36, gerakkan slider untuk melihat kesesuaian hasil baca sensor dengan di LCD dengan keterangan di sensor.
- **Untuk rangkaian berbasis hardware:** Gunakan es dan air hangat untuk melihat perubahan suhu secara signifikan.

2) Latihan

- Modifikasi rangkaian dan program yang ada, tambahkan output berupa LED RGB atau 3 buah LED secara terpisah.
- Buatlah rangkaian beserta program dengan memanfaatkan sensor selain TMP36. Silakan berkreativitas menggunakan berbagai sensor dan output yang tersedia.

4.7 PERCOBAAN 7 – SENSOR ULTRASONIK

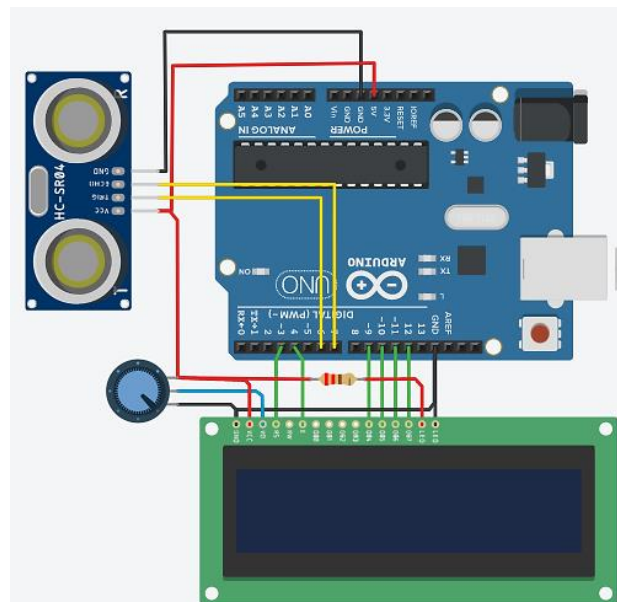
a. Tujuan Percobaan

- 1) Mengetahui prinsip kerja sensor ultrasonik
- 2) Membaca nilai sensor dan menampilkannya pada LCD
- 3) Memprogram Arduino yang terhubung dengan sensor dengan dan tanpa library

b. Alat dan Bahan

- 1) Arduino Uno
- 2) Sensor ultrasonik HC-SR04
- 3) LCD 16x2
- 4) Potensiometer
- 5) Resistor 220 Ω
- 6) Beberapa kabel penghubung

c. Rangkaian



Gambar 4.7. Rangkaian Percobaan 7

d. Program / Sketch

```
#include <LiquidCrystal.h>
const int RS=3, EN=4, D4=9, D5=10, D6=11, D7=12;
LiquidCrystal lcd (RS, EN, D4, D5, D6, D7);

//Inisiasi sensor ultrasonik
const int trig = 6; // Transmitter
const int echo = 7; // Receiver
float durasi, jarak;

void setup(){
  pinMode (trig, OUTPUT);
  pinMode (echo, INPUT);

  lcd.clear();
  lcd.begin (16,2);
  lcd.setCursor (0,0);
  lcd.print ("Jarak Objek");
}

void loop(){
  digitalWrite (trig, LOW);
  delay (2);
  digitalWrite (trig, HIGH);
  delayMicroseconds (10);
  digitalWrite(trig, LOW);

  durasi = pulseIn (echo, HIGH);
  jarak = durasi*17/1000;

  Serial.print("Jarak Objek : ");
  Serial.println(jarak);

  lcd.setCursor (0,1);
  lcd.print(jarak);
  lcd.print( "cm");
}
```

e. Instruksi dan Latihan

1) Instruksi

- Buatlah rangkaian dan program sesuai poin c dan d, jalankan program.
- Tekan sensor ultrasonik, geser objek, dan bandingkan nilai yang terbaca pada sensor dan LCD.

- Program pada poin **d** dibuat tanpa menggunakan library karena di Tinkercad belum mendukung penggunaan library HC-SR04. Oleh karena itu, jelaskan proses perhitungan jarak pada sensor ultrasonik berdasarkan program.
- Berikan penjelasan mengenai selisih jarak yang terbaca pada sensor dan LCD.
- **Untuk rangkaian berbasis hardware:** Gunakan sebuah benda/objek, dekatkan dan jauhkan dari sensor untuk melihat hasil baca sensor. Selanjutnya jelaskan proses perhitungan jarak pada sensor berdasarkan program.

2) Latihan

Buatlah sebuah rangkaian lain dengan sensor yang berbeda, kemudian ketikkan program dengan menggunakan library maupun tanpa library.

4.8 PERCOBAAN 8 – CONTOH PROJECT SEDERHANA

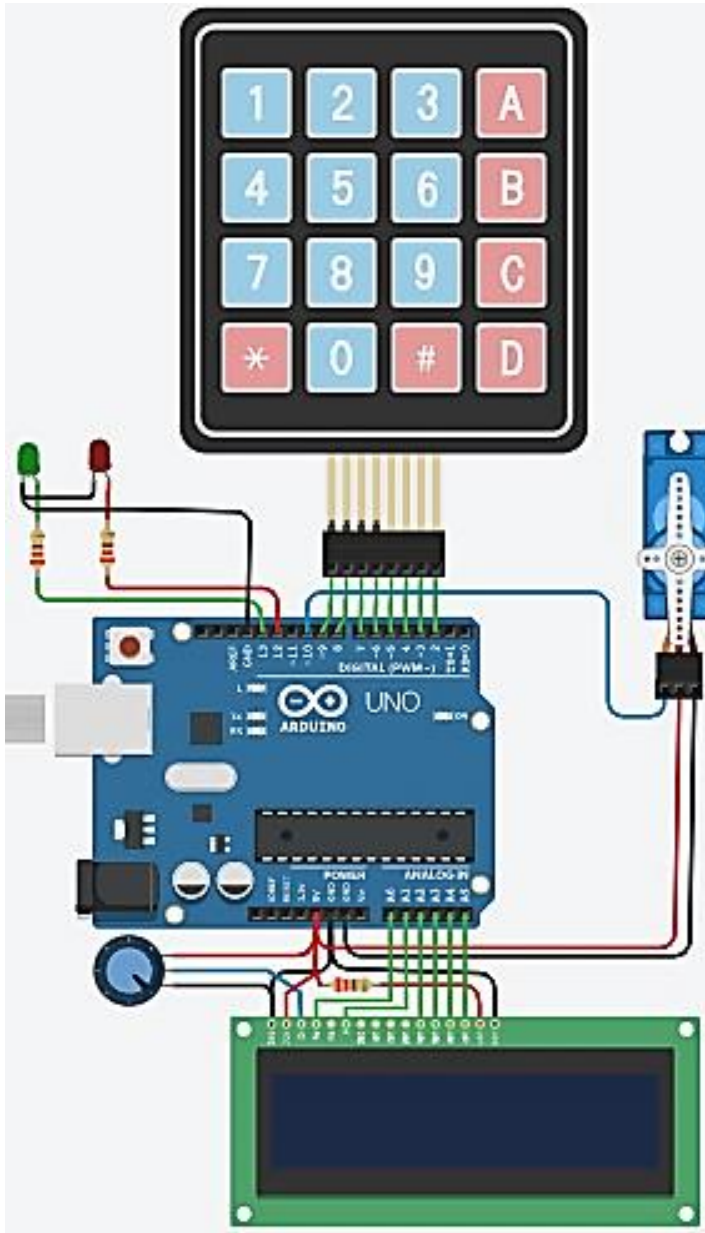
a. Tujuan Percobaan

- 1) Menerapkan penggunaan keypad, LCD, LED, dan motor servo
- 2) Membuat *project* sederhana untuk membuka pintu menggunakan *password*

b. Alat dan Bahan

- 1) Arduino Uno
- 2) Keypad
- 3) LCD 16x2
- 4) 2 buah LED
- 5) Potensiometer
- 6) 3 buah Resistor 220 Ω
- 7) Servo
- 8) Beberapa kabel penghubung

c. Rangkaian



Gambar 4.8. Rangkaian Percobaan 8

d. Program / Sketch

```
#include <LiquidCrystal.h> // Memasukkan library LCD
const int RS=A0, EN=A1, D4=A2, D5=A3, D6=A4, D7=A5;
LiquidCrystal lcd (RS, EN, D4, D5, D6, D7);

#include <Servo.h> // Memasukkan library Servo
Servo servo;

#include <Keypad.h> // Memasukkan library Keypad
const byte BARIS = 4; //empat baris
const byte KOLOM = 4; //empat kolom

//definisi simbol pada tombol yang ada di keypad
char petaTombol[BARIS][KOLOM] = {
    {'1','2','3','A'},
    {'4','5','6','B'},
    {'7','8','9','C'},
    {'*','0','#','D'}
};
byte pinBaris[BARIS] = {9, 8, 7, 6}; // Terhubung
// dengan pin
// baris pada
// keypad
byte pinKolom[KOLOM] = {5, 4, 3, 2}; // Terhubung
// dengan pin
// kolom pada
// keypad
Keypad tombol = Keypad(makeKeymap(petaTombol),
pinBaris, pinKolom, BARIS, KOLOM);
int ledM = 12, ledH = 13;
int posisi = 0;
char* password = "1234";

void setup(){
    lcd.begin(16,2);
    servo.attach(10);
    pinMode(ledH, OUTPUT);
    pinMode(ledM, OUTPUT);
    statusLock (true); // Status password
}

void loop(){

//Mendefinisikan tombol yang ditekan
char tombolDitekan = tombol.getKey();
lcd.setCursor (0,0);
lcd.print("SELAMAT DATANG");
lcd.setCursor(0,1);
lcd.print("Masukkan Psw");
```

```

        if(tombolDitekan == '*' || tombolDitekan == '#')
        ||
            tombolDitekan == 'A' || tombolDitekan == 'B' ||
            tombolDitekan == 'C' || tombolDitekan == 'D')
        {
            posisi = 0;
            statusLock (true);
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("    Psw Salah!    ");
            delay(100);
            lcd.clear();
            statusLock (true); // Untuk mengunci kembali
                                // setelah terbuka, tekan
                                // salah satu di antara
                                // *, #, A, B, C, D
        }
        if (tombolDitekan == password [posisi])
        {
            posisi ++;
        }
        if(posisi == 4)
        {
            statusLock (false);
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("*Terverifikasi*");
            delay(3000);
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("Selamat, Anda");
            lcd.setCursor(0, 1);
            lcd.print("Berhasil Masuk");
            delay(7000);
            lcd.clear();
        }
        delay(100);
    }

//Fungsi untuk mendefinisikan statusLock
void statusLock(int terkunci){
    if(terkunci){ // Jika terkunci LED Merah menyala
        digitalWrite(ledM, HIGH);
        digitalWrite(ledH, LOW);
        servo.write(90);
    }
    else{
        digitalWrite(ledM, LOW); // Jika tidak terkunci
                                // LED Hijau menyala
    }
}

```

```
digitalWrite(ledH, HIGH);  
servo.write(0);  
}  
}
```

e. Instruksi dan Latihan

1) Instruksi

- Buatlah rangkaian dan program sesuai poin **c** dan **d**, jalankan program.
- Tekan tombol sesuai password yang tersedia di program, pastikan rangkaian dan program telah berjalan dengan baik.

2) Latihan

Buatlah sebuah project sederhana menggunakan Mikrokontroler Arduino yang disertai dengan berbagai jenis sensor, display, dan komponen lainnya. Silakan berkreasi.

REFERENSI

Smith, G. A.(2011): Introduction to Arduino - A piece of cake.
ISBN: 1463698348.

Evans, B. W. (2008): Arduino Programming Notebook. ISBN:
978-1-4302-3778-5.

Purdum, J. (2011): Beginning C for Arduino. ISBN: 978-1-4302-
4777-7.

Crisp J. (2004): Introduction Microprocessors and
Microcontrollers (2nd Edition) - an imprint of Elsevier.
ISBN:0-7506-5989-0.

<https://e-dokumen.id/dokumen/>

[https:// id.wikipedia.org](https://id.wikipedia.org)

<https://www.arduino.cc>

<https://www.labelektronika.com>

<https://www.myusro.id>

<https://www.thinkercad.com>

<https://www.tptumetro.com>

<httpd://www.tutorialspoint.com>



Penulis bernama lengkap Ahmad Zarkasi dilahirkan di Selanglet pada tanggal 23 April 1991. Penulis menempuh pendidikan S1 Fisika di Universitas Mataram dan pendidikan S2 Fisika di Universitas Brawijaya dengan bidang minat Elektronika dan Instrumentasi. Saat ini, penulis merupakan staf pengajar di Program Studi Fisika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Mulawarman. Beberapa mata kuliah yang penulis ampu adalah Pengantar Mikrokontroler, Sistem Sensor, Fisika Instrumentasi, Instrumentasi Geofisika, Listrik Magnet, dan Fisika Eksperimen.

Tinkercad merupakan *platform* simulasi berbasis website yang fleksibel dan mudah digunakan. Salah satu fitur utama pada Tinkercad adalah fitur **circuits** yang dapat dijadikan sebagai media simulasi alternatif untuk keperluan pembelajaran mikrokontroler, khususnya mikrokontroler Arduino. Karena berbasis website, Tinkercad juga dapat dijadikan media pembelajaran *online* layaknya Edmodo, Schoology, dan Google Classroom, serta tentu saja dapat dengan mudah diakses melalui komputer maupun perangkat lain seperti *smartphone* dan *tablet*. Buku ini mencoba memperkenalkan *platform* Tinkercad berikut uraian rinci mengenai langkah-langkah penggunaannya. Selain itu, buku ini dilengkapi dengan beberapa *syntax* dasar dalam pemrograman Arduino.