

LAPORAN TUGAS SISTEM MIKROPROSESOR



**KELAS SISTEM MIKROPROSESOR B
UNIVERSAL SYNCHRONOUS AND ASYNCHRONOUS RECEIVER
TRANSMITTER (USART)**

Kelompok 2

**Muhammad Mukhlis Ikhsandy
Rizky Wulandari**

**04161045
04161067**

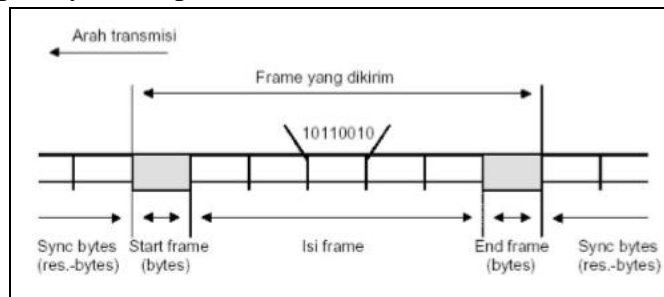
**INSTITUT TEKNOLOGIKALIMATAN
TEKNIK ELEKTRO
2018**

BAB I

Landasan Teori

1.1 Komunikasi Serial Synchronous dan Asynchronous

Komunikasi data serial mengenal dua buah metode, yaitu synchronous dan asynchronous. Metode synchronous mengirimkan datanya beberapa byte atau karakter (atau disebut blok data atau frame) sebelum meminta konfirmasi apakah data sudah diterima dengan baik atau tidak. Sementara metode asynchronous data dikirim satu byte setiap pengiriman. Biasanya tidak dibutuhkan konfirmasi penerimaan data. Dari kedua jenis metode tersebut dapat dipilih dan dilakukan lewat program. Tentu saja dibutuhkan program yang baik dan teliti untuk melakukannya. Namun dewasa ini proses pengiriman data serial tersebut sudah dilakukan oleh sebuah chip tersendiri (Hardware). Salah satu chip disebut UART (Universal Asynchronous Receiver Transmitter) dan satunya lagi disebut USART (Universal Synchronous Asynchronous Receiver Transmitter). Dalam protokol berbeda, synchronous memerlukan sinyal tambahan yang digunakan untuk mensinkronisasi setiap denyut dari proses transfer[1].



Gambar 1.1 Sinyal Transmisi Sinkron

Komunikasi data serial Asynchronous sekarang sudah digunakan demikian luas untuk transmisi yang berorientasi karakter, sementara metode Synchronous digunakan untuk transmisi yang berorientasi blok. Pada mode Asynchronous, setiap karakter ditempatkan berada diantara bit start dan bit stop. Bit start selalu satu bit, tapi stop bit bisa satu bit atau dua bit. Start bit selalu 0 (low) dan stop bit selalu 1 (high)[1].

Pada system yang sangat mementingkan integritas data yang disimpan, maka ditambahkanlah bit paritas kepada bingkai data tersebut. Maksudnya untuk setiap karakter 8-bit kita masih menambahkan bit paritas disamping bit start dan bit stop. Sehingga total adalah 11-bit. Adapun bit paritas adalah bit yang menunjukkan bahwa data yang dimaksud adalah memiliki jumlah bit 1s (high) ganjil atau genap. Bit paritas adalah bit di luar data yang bersangkutan atau merupakan tambahan. Chip UART khusus biasanya sudah dilengkapi dengan

keperluan paritas tersebut secara hardware. Bahkan ada beberapa pilihan untuk penanganan paritas ini, misalnya odd-, even- dan no-parity[1].

1.2 Arduino

Arduino adalah pengendali mikro single-board yang bersifat open-source, dirancang untuk memudahkan penggunaan elektronik dalam berbagai bidang. Hardware dalam arduino memiliki prosesor Atmel AVR dan menggunakan software dan bahasa sendiri[2].

1.2.1 Hardware

Hardware dalam arduino memiliki beberapa jenis, yang mempunyai kelebihan dan kekurangan dalam setiap papannya. Penggunaan jenis arduino disesuaikan dengan kebutuhan, hal ini yang akan mempengaruhi dari jenis prosesor yang digunakan. Jika semakin kompleks perancangan dan program yang dibuat, maka harus sesuai pula jenis kontroler yang digunakan. Yang membedakan antara arduino yang satu dengan yang lainnya adalah penambahan fungsi dalam setiap boardnya dan jenis mikrokontroler yang digunakan. Dalam laporan ini, jenis arduino yang digunakan adalah arduino uno[2].

1.2.1.a Arduino Uno

Arduino Uno adalah salah satu produk berlabel arduino yang sebenarnya adalah suatu papan elektronik yang mengandung mikrokontroler ATmega328 (sebuah keping yang secara fungsional bertindak seperti sebuah komputer). Piranti ini dapat dimanfaatkan untuk mewujudkan rangkaian elektronik dari yang sederhana hingga yang kompleks. Pengendalian LED hingga pengontrolan robot dapat diimplementasikan dengan menggunakan papan berukuran relatif kecil ini. Bahkan dengan penambahan komponen tertentu, piranti ini bisa dipakai untuk pemantauan kondisi pasien di rumah sakit dan pengendalian alat-alat di rumah[2].

Tabel 1.1 Index Board Arduino

Mikrokontroler	ATmega328
Tegangan pengoperasian	5V
Tegangan Input yang disarankan	7-12V
Batas tegangan input	6-20V
Jumlah pin I/O digital	14 (6 diantaranya menyediakan keluaran PWM)
Jumlah pin input analog	6
Arus DC tiap pin I/O	40 mA
Arus DC untuk pin 3.3V	50mA
Memori Flash	32 KB (ATmega328), sekitar 0.5

	KB digunakan oleh bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz



Gambar 1.2 Arduino Uno

Hardware arduino uno memiliki spesifikasi sebagai berikut:

- a. 14 pin IO Digital (pin 0–13) Sejumlah pin digital dengan nomor 0–13 yang dapat dijadikan input atau output yang diatur dengan cara membuat program IDE.
- b. 6 pin Input Analog (pin 0–5) Sejumlah pin analog bernomor 0–5 yang dapat digunakan untuk membaca nilai input yang memiliki nilai analog dan mengubahnya ke dalam angka antara 0 dan 1023.
- c. 6 pin Output Analog (pin 3, 5, 6, 9, 10 dan 11) Sejumlah pin yang sebenarnya merupakan pin digital tetapi sejumlah pin tersebut dapat diprogram kembali menjadi pin output analog dengan cara membuat programnya pada IDE. Papan Arduino Uno dapat mengambil daya dari USB port pada komputer dengan menggunakan USB charger atau dapat pula mengambil daya dengan menggunakan suatu AC adapter dengan tegangan 9 volt. Jika tidak terdapat power supply yang melalui AC adapter, maka papan Arduino akan mengambil daya dari USB port. Tetapi apabila diberikan daya melalui AC adapter secara bersamaan dengan USB port maka papan Arduino akan mengambil daya melalui AC adapter secara otomatis[2].

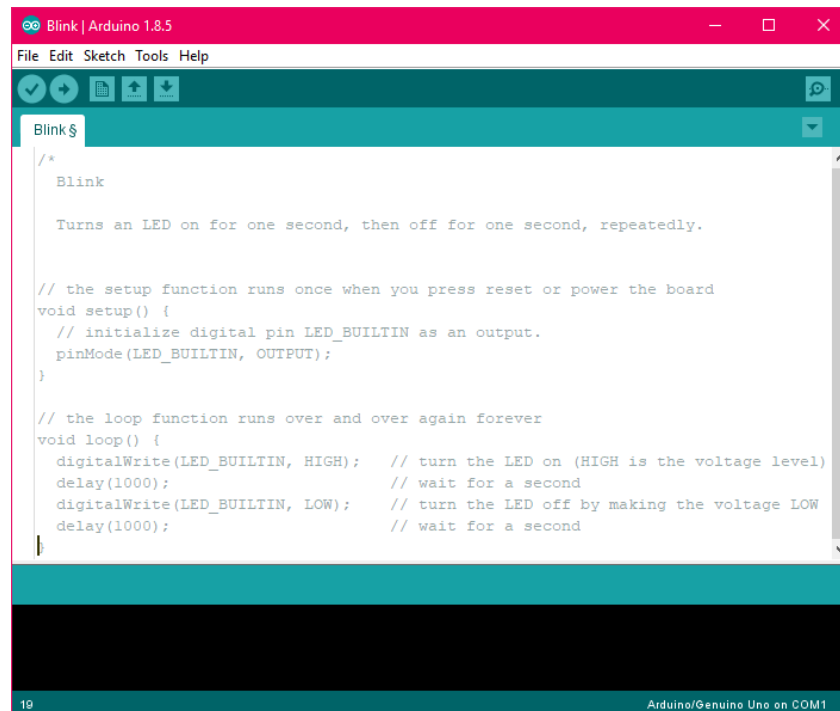
1.2.2 Software

Software arduino yang digunakan adalah driver dan IDE, walaupun masih ada beberapa software lain yang sangat berguna selama pengembangan arduino.

Integrated Development Environment (IDE), suatu program khusus untuk suatu komputer agar dapat membuat suatu rancangan atau sketsa program untuk papan Arduino. IDE arduino merupakan software yang sangat canggih ditulis dengan menggunakan java. IDE arduino terdiri dari :

1. Editor Program Sebuah window yang memungkinkan pengguna menulis dan mengedit program dalam bahasa processing.
2. Compiler Berfungsi untuk kompilasi sketch tanpa unggah ke board bisa dipakai untuk pengecekan kesalahan kode sintaks sketch. Sebuah modul yang mengubah kode program menjadi kode biner bagaimanapun sebuah mikrokontroler tidak akan bisa memahami bahasa processing.
3. Uploader Berfungsi untuk mengunggah hasil kompilasi sketch ke board target. Pesan error akan terlihat jika board belum terpasang atau alamat port COM belum terkonfigurasi dengan benar. Sebuah modul yang memuat kode biner dari komputer ke dalam memory didalam papan arduino.

1.3 Arduino IDE



Gambar 1.3 Tampilan Home Arduino IDE

Kode Program Arduino biasa disebut sketch dan dibuat menggunakan bahasa pemrograman C. Program atau sketch yang sudah selesai ditulis di Arduino IDE bisa langsung dicompile dan diupload ke Arduino Board. Secara sederhana,

sketch dalam Arduino dikelompokkan menjadi 3 blok (lihat gambar di atas): 1. Header 2. Setup 3. Loop

1. Header Pada bagian ini biasanya ditulis definisi-definisi penting yang akan digunakan selanjutnya dalam program, misalnya penggunaan library dan pendefinisian variable. Code dalam blok ini dijalankan hanya sekali pada waktu compile. Di bawah ini contoh code untuk mendeklarasikan variable led (integer) dan sekaligus di isi dengan angka 13 `int led = 13;` 2. Setup Di sinilah awal program Arduino berjalan, yaitu di saat awal, atau ketika power on Arduino board. Biasanya di blok ini diisi penentuan apakah suatu pin digunakan sebagai input atau output, menggunakan perintah `pinMode`. Inisialisasi variable juga bisa dilakukan di blok ini // the setup routine runs once when you press reset: `void setup() { // initialize the digital pin as an output. pinMode(led, OUTPUT); }` OUTPUT adalah suatu makro yang sudah didefinisikan Arduino yang berarti = 1. Jadi perintah di atas sama dengan `pinMode(led, 1);`

Suatu pin bisa difungsikan sebagai OUTPUT atau INPUT. Jika difungsikan sebagai output, dia siap mengirimkan arus listrik (maksimum 100 mA) kepada beban yang disambungkannya. Jika difungsikan sebagai INPUT, pin tersebut memiliki impedance yang tinggi dan siap menerima arus yang dikirimkan kepadanya.

3. Loop Blok ini akan dieksekusi secara terus menerus. Apabila program sudah sampai akhir blok, maka akan dilanjutkan dengan mengulang eksekusi dari awal blok. Program akan berhenti apabila tombol power Arduino di matikan. Di sinilah fungsi utama program Arduino kita berada. `void loop() { digitalWrite(led, HIGH); // nyalakan LED delay(1000); // tunggu 1000 milidetik digitalWrite(led, LOW); // matikan LED delay(1000); // tunggu 1000 milidetik }` Perintah `digitalWrite(pinNumber,nilai)` akan memerintahkan arduino untuk menyalakan atau mematikan tegangan di pinNumber tergantung nilainya. Jadi perintah di atas `digitalWrite(led,HIGH)` akan membuat pin nomor 13 (karena di header dideklarasikan `led = 13`) memiliki tegangan = 5V (HIGH). Hanya ada dua kemungkinan nilai `digitalWrite` yaitu HIGH atau LOW yang sebetulnya adalah nilai integer 1 atau 0. Kalau sudah dibuat program diatas, selanjutnya kita ambil kabel USB yang diikutsertakan pada saat membeli Arduino, pasangkan ke komputer dan board arduino, dan upload programnya. Lampu LED yg ada di Arduino board kita akan kelap-kelip. Sekedar informasi, sebuah LED telah disediakan di board Arduino Uno dan disambungkan ke pin 13. Selain blok `setup()` dan `loop()` di atas kita bisa mendefinisikan sendiri blok fungsi sesuai kebutuhan. Kita akan jumpai nanti pada saat pembahasan proyek[2].

1.4 USART

USART (Universal Synchronous Asynchronous Received Transmitter) merupakan salah satu mode komunikasi yang dimiliki oleh Mikrokontroler ATmega8535. USART memiliki 2 pin (RxD dan TxD) untuk Asynchronous dan 3 bit TxD, RxD, xCK untuk Synchronous[3].

Untuk mengatur komunikasi USART dilakukan melalui beberapa register yaitu : UDR (USART Data Register) adalah register yang paling penting dalam komunikasi serial ini. Sebab data yang dikirim keluar harus ditempatkan pada register ini, sedang data yang diterima dari luar dapat dibaca pada register ini pula. Pada intinya register UDR digunakan sebagai buffer untuk menyimpan data, baik yang akan dikirim maupun yang akan diterima[3].

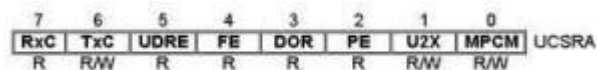


Gambar 1.4 UDR

Seperti yang dapat dilihat pada gambar, sejatinya UDR adalah terdiri dari 2 buah register terpisah, dengan alamat dan nama yang sama, yakni UDR. Saat kita menulis data pada UDR ini, maka sebenarnya kita menulis data pada UDR (Write) yang kemudian USART mem-frame dengan bit-bit frame dan segera akan segera mengirimkan data tersebut secara serial. Saat kita membaca UDR, sebenarnya adalah membaca UDR (Read). Data yang diterima secara serial akan disimpan dalam register tersebut, setelah hadirnya stop bit, maka USART akan membuang frame dan menyiapkan data pada UDR (Read) sehingga dapat segera di ambil. Kita dapat menggunakan instruksi OUT dan IN untuk menulis dan membaca register UDR ini.

UCSRA (USART Control and Status Register A)

adalah register yang penting. Sebagian besar adalah berisi status dari dari proses transfer komunikasi serial itu sendiri. Adapun penjelasan dari bit-bit tersebut adalah:



Gambar 1.5 Bit UCSRA

Bit 7 – RxC: USART Receive Complete

Bit ini menjadi tinggi jika ada data yang masih belum diambil atau dibaca di dalam buffer penerima (UDR-read). Bit ini akan otomatis rendah setelah buffer penerima telah dibaca. Jika Unit Penerima tiba-tiba dimatikan setelah diaktifkan, maka isi dalam buffer penerima akan langsung dibuang (flushed) dan bit RxC ini akan langsung dibuat rendah. Bit ini juga bisa mengaktifkan instruksi “Receive Complete

interrupt ”. Lihat penjelasan tentang bit RxCIE. Untuk mengetahui penerimaan data jika high (1) ada data baru dan jika low (0) tidak ada data baru.

Bit 6 – TxC: USART Transmit Complete

Bit ini akan otomatis tinggi saat semua frame dalam shift-register pengiriman telah digeser semuanya keluar dan jika tidak ada data baru yang berarada dalam buffer pengiriman (UDR-write). Bit TxC ini akan otomatis rendah setelah “Transmit Complete interrupt ” dijalankan, atau dengan meng-clearkan secara manual dengan cara menulis bit ini dengan nilai 1’s (tinggi). Bit TxC ini pula dapat membangkitkan “Transmit Complete interrupt ”. Lihat penjelasan tentang bit TxCIE. Untuk mengetahui pengiriman data, jika high (1) ada data baru dan jika low (0) tidak ada data baru.

Bit 5 – UDRE: USART Data Register Empty

Bit UDRE ini adalah untuk menjadikan tanda jika buffer pengiriman (UDR-write) telah siap untuk diberikan data baru. Bit ini akan bernilai 1 (tinggi) , berarti kita saat itu boleh menulis UDR. Bit ini dapat membangkitkan UDRIE atau “Data Register Empty interrupt ”. Lihat penjelasan tentang bit UDRIE. Bit ini setelah reset langsung bernilai 1, yang berarti siap untuk melakukan pengiriman.

Bit 4 – FE: Frame Error

Bit ini otomatis menjadi tinggi jika saat menerima data, ternyata ada kesalahan dari frame yang diterima. Misalnya saat Unit penerima seharusnya menunggu sebuah bit Stop, ternyata data yang ada adalah 0 (rendah). Bit ini valid setelah kita membaca UDR. Harap selalu menulis bit ini dengan 0 (rendah) saat kita sedang menulis UCSRA.

Bit 3 – DOR: Data OverRun

Bit ini akan menjadi tinggi saat kondisi overrun terjadi. Kondisi ini terjadi saat buffer penerima sudah penuh dan berisi 2 data karakter, dimana data karakter terakhir tidak bisa dipindahkan ke UDR-read, karena tidak kunjung dibaca oleh user. Bit ini valid setelah kita membaca UDR. Harap selalu menulis bit ini dengan 0 (rendah) saat kita sedang menulis UCSRA.

Bit 2 – PE: Parity Error

Bit ini akan menjadi tinggi saat karakter yang sedang diterima ternyata memiliki format parity yang salah. Tentu saja hal ini terjadi jika bit parity checking diaktifkan (UPM1 = 1). Bit ini valid setelah kita membaca UDR. Harap selalu menulis bit ini dengan 0 (rendah) saat kita sedang menulis UCSRA.

Bit 1 – U2X: Double the USART Transmission Speed

Bit ini hanya berlaku untuk operasi tak sinkron (asynchronous). Jika bit ini kita tulis dengan 1’s (tinggi) maka baud rate akan menjadi lebih cepat 2 kali. Hal itu terjadi karena pembagi baud rate yang biasanya membagi 16 kemudian membagi menjadi dengan 8 saja. Tulis bit ini dengan 0’s (rendah) untuk oprasi sinkron (synchronous).

Bit 0 – MPCM: Multi-processor Communication Mode

Bit ini digunakan untuk mode komunikasi Multi-Prosesor. Saat bit PMCM ini dibuat menjadi tinggi maka setiap data yang diterima oleh unit penerima, namun tidak dilengkapi dengan informasi alamat, data yang benar, maka akan diabaikan. Bit ini hanya berguna untuk penerima, dan bukan untuk pengirim.

UCSRC (USART Control dan Status Register C)

regsiter ini adalah register penting, untuk melakukan kontrol pada peralatan USART. Namuin jika tidak menggunakan fungsi USART, maka boleh mengabaikan register ini seperti dalam keadaan resetnya. dengan nilai \$96, yang berarti bahwa sedang meggunakannya sebagai UART, Parity-none, 1 stop bit, ukuran data 8-bit (UCSZ2 = 0).

7	6	5	4	3	2	1	0	
URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
1	0	0	0	0	1	1	0	

Gambar 1.6 UCSRC

Catatan : Register UCSRC adalah berbagi alamat I/O yang sama dengan register UBBRH. Lihat bagaimana mengakses UBBRH dan UCSRC pada “Accessing UBBRH/UCSRC Registers” untuk mendapatkan penjelasan lebih lengkap.

Bit 7 – URSEL: Register Select

Saat akan menulis port \$20 maka ada dua register yang akan akan diakses, yang ditentukan dari D7 dari data yang dituliskan. Jika D7 (MSB) adalah 1 atau datanya diatas \$80 maka sedang menuliskan data pada UCSRC. Sedang jika data yang hendak dituliskan adalah dibawah \$80, maka kita sedang menulis UBBRH. Bagaimana saat membaca UCSRC, pastikan sebelumnya bit ini dalam keadaan tinggi sebelumnya membaca lokasi \$20 ini.

Bit 6 – UMSEL: USART Mode Select

Bit ini untuk memilih USART dijadikan mode Taksinkron (Asynchronous) atau Sinkron (Synchronous).

UMSEL	MODE
0	Asynchronous Operation
1	Synchronous Operation

Gambar 1.7 UMSEL

Bit 5:4 – UPM1:0: Parity Mode

Bit-bit ini adalah untuk menghidupkan pembangkit dan pemeriksa parity. Jika diaktifkan maka akan otomatis membangkitkan parity pada setiap data yang dikirimkan dan akan memeriksa validitas parity dari setiap data yang diterima. Pada Unit penerima kita akan mendapatkan parity yang diterima, kemudian parity tersebut akan dibandingkan dengan status dari UMP0. Jika ternyata tidak cocok, maka bendera bit PE (Parity Error) pada UCSRA akan diaktifkan.

UPM1	UPM0	PARITY MODE
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

Gambar 1.8 Tabel Pengaturan Parity

Bit 3 – USBS: Stop Bit Select

Dengan membiarkan bit ini menjadi 0 maka frame akan dilengkapi dengan Stop-bit selebar 1-bit. Sedang jika bit ini ditulis tinggi, maka bit stop menjadi 2-bit. Stop bit sepanjang 2-bit ini biasanya diperlukan bagi system lain yang terhubung dengan AVR akan memiliki waktu yang cukup untuk memproses data yang baru saja diterimanya, dan sudah benar-benar siap untuk menerima data berikutnya. Unit penerima USART tidak menggunakan bit ini.

USBS	STOP BIT (s)
0	1-Bit
1	2-Bit

Gambar 1.9 Tabel Jumlah Bit sebagai Bit Stop

Bit 2:1 – UCSZ1:0: Character Size

Menentukan karakter dari data yang hendak dikirimkan dan diterima haruslah merujuk pada bit-bit ini. Yakni bit UCSZ1 dan bit UCSZ0 milik register ini. Ditambah dengan bit UCSZ2 pada register UCSRB.

UCSZ2	UCSZ1	UCSZ0	CHARACTER SIZE
0	0	0	5-Bit
0	0	1	6-Bit
0	1	0	7-Bit
0	1	1	8-Bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-Bit

Gambar 1.10 Tabel Jumlah Data Yang di Transfer

Bit 0 – UCPOL: Clock Polarity

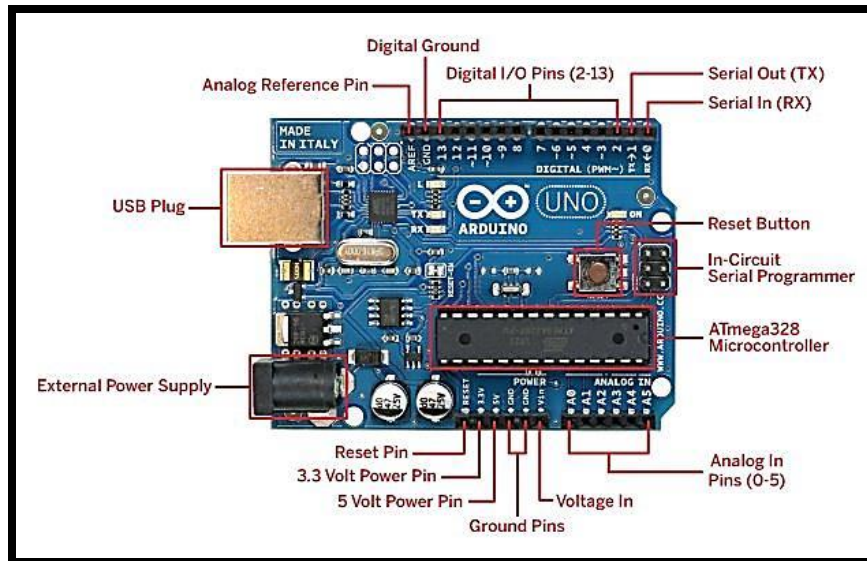
Bit ini hanya digunakan pada mode Synchronous. Dalam mode ini akan direlasikan antara daya yang diterima dan data yang dikirm dan disinkronkan dengan status dari XCX (synchronous clock). Perhatikan tabel 5 untuk mendalami bagaimana sinkronikasi terjadi untuk ujung XCX (edge) yang berbeda dengan mengatur bit UCPOL ini[3].

UCPOL	Transmitted Data Changed (Output of TxD Pin)	Received Data Sampled (Input on RxD Pin)
0	Falling xCK Edge	Rising xCK Edge
1	Rising xCK Edge	Falling xCK Edge

Gambar 1.11 Tabel Pengaturan Polaritas Dalam Mode Synchronous

BAB II Pembahasan

2.1 Hardware

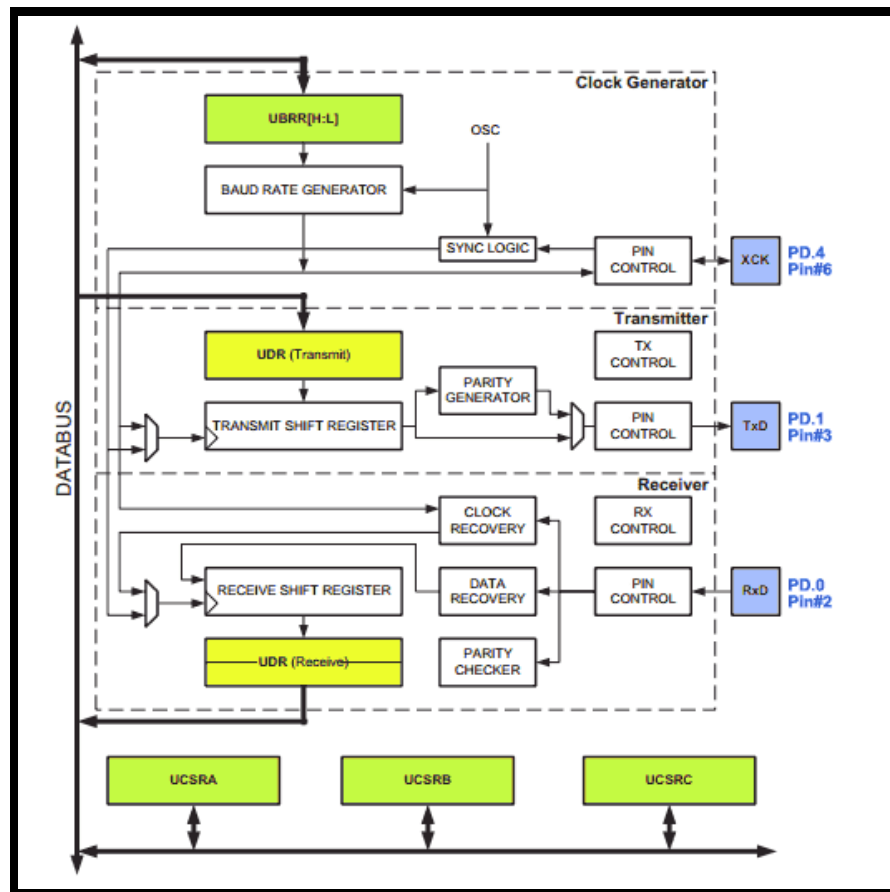


Gambar 2.1 Bagian-bagian pada Arduino ATmega328

(PCINT14/RESET) PC6	1	28	PC5 (ADC5/SCL/PCINT13)
(PCINT16/RXD) PD0	2	27	PC4 (ADC4/SDA/PCINT12)
(PCINT17/TXD) PD1	3	26	PC3 (ADC3/PCINT11)
(PCINT18/INT0) PD2	4	25	PC2 (ADC2/PCINT10)
(PCINT19/OC2B/INT1) PD3	5	24	PC1 (ADC1/PCINT9)
(PCINT20/XCK/T0) PD4	6	23	PC0 (ADC0/PCINT8)
VCC	7	22	GND
GND	8	21	AREF
(PCINT6/XTAL1/TOSC1) PB6	9	20	AVCC
(PCINT7/XTAL2/TOSC2) PB7	10	19	PB5 (SCK/PCINT5)
(PCINT21/OC0B/T1) PD5	11	18	PB4 (MISO/PCINT4)
(PCINT22/OC0A/AIN0) PD6	12	17	PB3 (MOSI/OC2A/PCINT3)
(PCINT23/AIN1) PD7	13	16	PB2 (SS/OC1B/PCINT2)
(PCINT0/CLKO/ICP1) PB0	14	15	PB1 (OC1A/PCINT1)

Gambar 2.2 Pin pada mikrokontroler ATmega328

Untuk mengakses pin USART pada Arduino ATmega328, digunakan pin Tx dan Rx. Kedua pin ini merupakan jalur data komunikasi serial dengan level sinyal TTL. Pin TXD berfungsi untuk mengirimkan data serial, sedangkan RXD kebalikannya yaitu sebagai pin yang berfungsi untuk menerima data serial. Selain kedua pin tersebut, terdapat pin clock eksternal berupa pin XCK dapat difungsikan sebagai sumber clock external untuk USART, namun kita juga dapat memanfaatkan clock dari CPU, sehingga tidak perlu membutuhkan external clock.



Gambar 2. Blok Diagram USART

Pada diagram blok USART di atas, garis tebal menunjukkan I/O dan register yang dapat diakses oleh CPU. Blok diagram di atas terbagi menjadi tiga bagian penting yaitu Clock Generator, Transmitter dan Receiver. Register Control (UCSRA, UCSRB dan UCSRC) dapat terhubung ke semua unit. Clock Generator terdiri dari Sync Logic sebagai input clock eksternal yang digunakan operasi sinkron Slave. Generator baud rate dan XCK hanya digunakan saat transfer dalam mode sinkron.

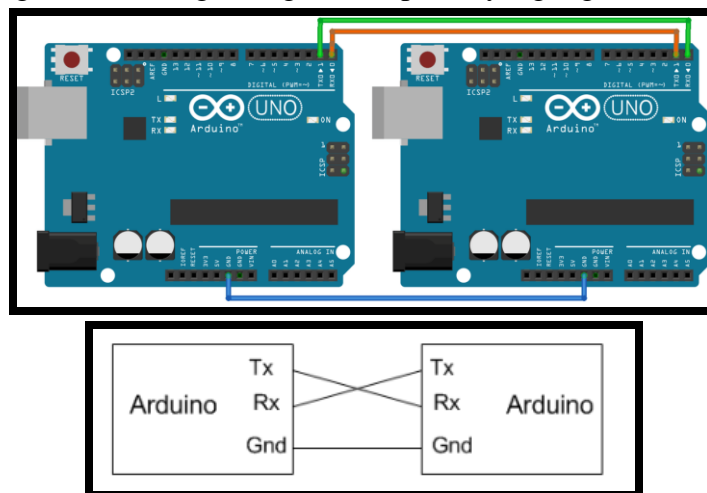
Transmitter terdiri dari single write buffer, serial shift register, parity generator dan control logic untuk menangani serial dengan format berbeda. Write buffer memungkinkan transfer data tanpa delay antar frame.

Receiver merupakan bagian paling kompleks karena Clock dan unit data recovery. Unit Recover digunakan untuk penerimaan data asinkron. Tambahan pada unit recover, receiver terdiri dari parity checker, logic control, shift register dan UDRn. Receiver dapat mendeteksi frame error, data overrun dan parity error.

Receiver mulai menerima data saat receiver mendeteksi start bit yang valid. Bit yang berada dibelakang start bit akan diuji di baud rate/XCK lalu digeser ke shift register receiver sampai stop bit pertama dari frame tersebut diterima. Ketika frame

tersebut telah lengkap di shift register, maka frame tersebut akan dipindahkan ke receiver buffer, Receiver Buffer dapat dibaca dengan membaca lokasi Input Output UDRn.

Transmitter dikatiffkan oleh UCSRB dan TxD akan aktif sebagai transmitter serial output.. Baud rate, mode operasi dan format frame harus diatur terlebih dahulu sebelum melakukan pengiriman. Jika mode operasi sinkron yang digunakan, clock di XCKn akan dikendalikan dan digunakan sebagai clock saat pengiriman. Sebuah transmisi data diinisiasi dengan memuat transmit buffer dengan data yang akan di kirim . CPU dapat memuat transmit buffer dengan menulisnya ke lokasi input output UDRn.Data yang berada didalam transmit buffer akan di pindahkan ke Shift Register ketika Shift Register siap untuk mengirim frame baru. Shift Register memuat data baru jika tidak ada pengiriman yang sedang berlangsung atau stop bit terakhir dari frame sebelumnya telah dikirim. Ketika shift register memuat data baru, shift register akan mengirimkan satu frame lengkap dalam kecepatan yang diberikan oleh baud rate, bit U2Xn atau dengan XCKn tergantung mode operasi yang digunakan.



Gambar 2.4 Komunikasi serial antar mikrokontroler menggunakan pin USART

Untuk komunikasi serial antar dua mikrokontroler atau lebih dengan menggunakan pin USART, maka pada pin TX pada Arduino1 dihubungkan pada pin Rx pada Arduino2 dan pin Rx pada Arduino1 dihubungkan pada pin Tx pada Arduino2. Dan pin GND kedua mikrokontroler saling dihubungkan. Konfigurasi ini dilakukan agar kedua mikrokontroler dapat saling mengirim (Transmit) dan menerima (Receive) data. Arduino1 akan mengirim data melalui pin Tx dan akan diterima Arduino2 akan menerima data melalui pin Rx. Data yang telah diproses oleh Arduino2 akan dikirim kembali melalui pin Tx menuju pin Rx pada Arduino1.

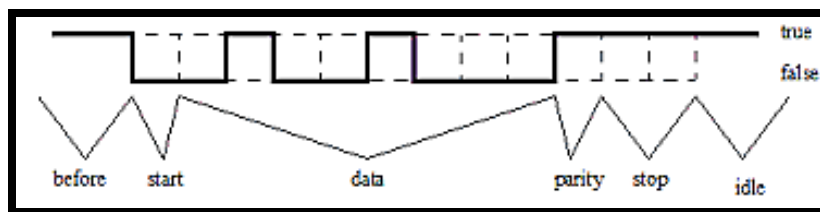
2.2 Cara Kerja USART

USART (Universal Syschronous Asynchronous Received Transmitter) merupakan jenis perangkat perangkat keras komunikasi seperti register yang memungkinkan komputer untuk berkomunikasi dua arah sinkron dan asinkron dengan perangkat yang terhubung secara serial.

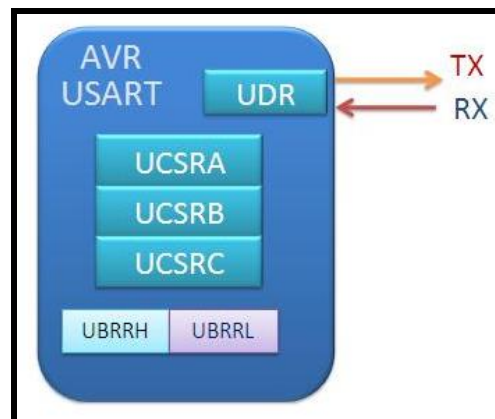
Pada transimi sinkron pengirim akan mengirimkan clock / timing signal sehingga device penerima tahu kapan membaca bit data berikutnya. Transimisi asinkron mengijinkan pengirim tidak memberikan clock sinyal pada penerima, sebagai gantinya untuk memulai transmisi pengirim mengirimkan start bit pada tiap byte data yang dikirimkan dan diakhiri dengan stop bit.

Komunikasi dengan menggunakan USART dapat dilakukan dengan dua cara yaitu:

1. Mode sinkron dimana pengirim data mengeluarkan pulsa/clock untuk sinkronisasi data
2. 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 ini setiap perangkat yang terhubung harus memiliki baud rate (laju data) yang sama.



Gambar 2.5 Mode komunikasi asinkron



Gambar 2.6 Register USART

1. Register Data (UDR), menyimpan data yg dikirim dan diterima.
 - Bit 5 – UDRIE: USART Data Register Empty Interrupt Enable
2. Register Control (UCSRA bit 0~ bit1, UCSRB dan UCSRC)
3. Register Status (UCSRA bit 2~bit 7)



Gambar 2.7 Register UCSRA

- Bit 7 – RXC: USART Receive Complete
RXC otomatis akan bernilai 1, jika ada data baru di bufer penerima. RXC otomatis akan bernilai 0, jika data sudah dibaca atau bufer penerima kosong.
- Bit 6 – TXC: USART Transmit Complete
TXC otomatis akan bernilai 1, jika data di buffer selesai dikirim.
- Bit 5 – UDRE: USART Data Register Empty
UDRE otomatis akan bernilai 1, jika register UDR kosong transmitter siap mengirim data. UDRE=0, UDR berisi data yg belum selesai dikirim.
- Bit 4 – FE: Frame Error
FE otomatis akan bernilai 1, jika ada frame error.
- Bit 3 – DOR: Data OverRun
DOR otomatis akan bernilai 1, jika data datang ketika bufer penuh(terjadi antrian).
- Bit 2 – PE: Parity Error
PE otomatis akan bernilai 1, jika terjadi parity error.
- Bit 1 – U2X: Double the USART Transmission Speed
kita set U2X=0, kecepatan normal. U2X=1 kecepatan 2xbaudrate.
- Bit 0 – MPCM: Multi-processor Communication Mode
kita set MCM=1 byte pertama yg diterima harus 9 bit, jika tdk data byte akan diabaikan. bit ini terjadi hanya untuk penerimaan saja pd komunikasi banyak microcontroller.



Gambar 2.8 Register UCSRB

- Bit 7 – RXCIE: RX Complete Interrupt Enable
kita set RXCIE=1, interupsi receive complete aktif.
- Bit 6 – TXCIE: TX Complete Interrupt Enable
kita set TXCIE=1, interupsi transmit complete aktif.
- Bit 5 – UDRIE: UDRE Interrupt Enable
kita set UDRIE=1, interupsi UDRE aktif.
- Bit 4 – RXEN: Receiver Enable
kita set RXEN=1, USART receiver aktif. mikrokontroler bisa menerima data.
- Bit 3 – TXEN: Transmitter Enable

kita set TXEN=1, Usart Transmitter aktif. mikrokontroler bisa mengirim data.

- Bit 2 – UCSZ2: Character Size

kita set UCSZ2:UCSZ1:UCSZ0 = 011 , panjang data 8 BIT. (bit UCSZ1 dan UCSZ0 ada di register UCSRC). Berikut tabel pengaturan bit pada UCSZ.

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

UCSZ Bits Settings

- Bit 1 – RXB8: Receive Data Bit 8

RXB8 menjadi bit ke-9 jika panjang data yg diterima 9 bit .

- Bit 0 – TXB8: Transmit Data Bit 8

TXB8 menjadi bit ke-9 jika panjang data yg dikirim 9 bit.



Gambar 2.9 Register UCSRC

- Bit 7 – URSEL: Register Select . memilih UCSRC atau UBRRH

kita set URSEL=1 , UCSRC aktif ,UBRRH tdk aktif,

kita set URSEL=0 , UBRRH aktif , UCRSC tdk aktif.

- Bit 6 – UMSEL: USART Mode Select

kita set UMSEL=1 , mode synceonous. UMSEL=0 mode asynceonous

- Bit 5:4 – UPM1:UMP0: Parity Mode

kita set :

UPM1	UPM0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

UPM Bits Settings

- Bit 3 – USBS: Stop Bit Select

kita set USBS=0, stop bit =1 bit , USBS=1 panjang stop bit = 2 bit.

- Bit 2:1 – UCSZ1:0: Character Size

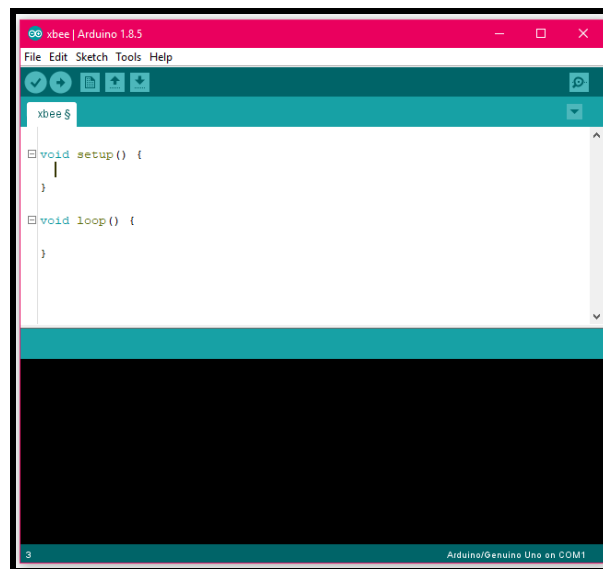
kita set UCSZ2:UCSZ1:UCSZ0 = 011 , panjang data 8 BIT. (bit UCSZ2 ada di register UCSRB)

- Bit 0 – UCPOL: Clock Polarity bit ini digunakan untuk mode syncronous saja.

kita set UCPOL=0 trnasmisi clock naik, UCPOL=1 transmisi clock turun. (khusus yg ini don't care krn kita menggunakan mode asynchronous)

4. Register 8 bit UBRRH dan 8 bit UBRRL , menyimpan parameter baudrate 16 bit UBRR register.

2.3 Software



Gambar 2.10 Interface Arduino IDE

IDE itu merupakan kependekan dari *Integrated Development Enviroenment*, atau secara bahasa mudahnya merupakan lingkungan terintegrasi yang digunakan untuk melakukan pengembangan. Disebut sebagai lingkungan karena melalui software inilah Arduino dilakukan pemrograman untuk melakukan fungsi-fungsi yang dibenamkan melalui sintaks pemrograman. Arduino IDE dibuat dari bahasa pemrograman JAVA. Arduino IDE juga dilengkapi dengan library C/C++ yang biasa disebut *Wiring* yang membuat operasi input dan output menjadi lebih mudah. Arduino IDE ini dikembangkan dari software **.Processing** yang dirombak menjadi Arduino IDE khusus untuk pemrograman dengan Arduino

Komunikasi data serial digunakan untuk komunikasi antara board arduino dengan komputer atau perangkat lain. Semua board arduino mempunyai sedikitnya 1 buah port serial yang juga dikenal dengan nama UART atau USART. Komunikasi data serial menggunakan 2 buah pin yaitu pin RX untuk menerima data dan ppin TX

untuk mengirimkan data. Pada board arduino pin RX terletak pada pin0 dan pin TX terletak pada pin1. Ketika board arduino dikonfigurasi untuk berkomunikasi secara serial, maka kedua pin0 dan pin1 tidak dapat digunakan sebagai pin input/output digital.

Program Arduino telah dilengkapi dengan serial port library yang memudahkan programmer untuk membuat program.

1. Serial.available()

```
int dataterkirim = 0;

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

void loop()
{
  if (Serial.available() > 0)
  {
    dataterkirim = Serial.read();
    Serial.print("saya menerima data: ");
    Serial.println(dataterkirim, DEC);
  }
}
```

Gambar 2.11 Serial available

Digunakan untuk menyatakan angka, bytes atau karakter yang sudah siap dibaca dari serial port. Data ini adalah data yang telah diterima dan disimpan dalam serial receive buffer. Serial receive buffer dapat menampung 64 bytes data. Berikut ini contoh penulisan instruksi available.

2. Serial.begin()

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

Gambar 2.12 Serial begin

Digunakan untuk mengatur baudrate / kecepatan transmisi data. Beberapa pilihan kecepatan komunikasi data yang dapat digunakan pada board arduino adalah 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 atau 115200. Pengaturan baudrate dilakukan pada bagian setup(). Berikut ini format penulisan dan contoh program “begin()”

3. Serial.end()

```
Serial.end
```

Gambar 2.13 Serial end

Digunakan untuk menutup komunikasi serial port. Berikut format penulisan dan contoh program. Untuk membuka kembali komunikasi serial port dapat menggunakan perintah “Serial.begin()”.

4. Serial.find()

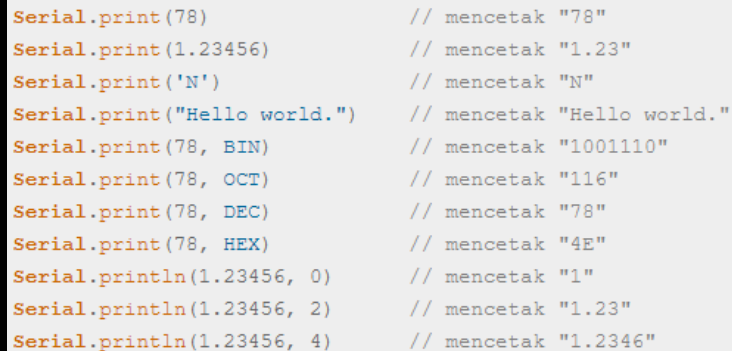


```
Serial.find
```

Gambar 2.14 Serial find

Perintah “Serial.find()” digunakan untuk membaca data dari serial port buffer hingga target yang ditentukan dalam perintah “Serial.find()” terpenuhi. Fungsi akan bernilai benar jika target tercapai dan bernilai salah jika target tidak tercapai. Berikut ini contoh penulisan program “Serial.find()”.

5. Serial.print()

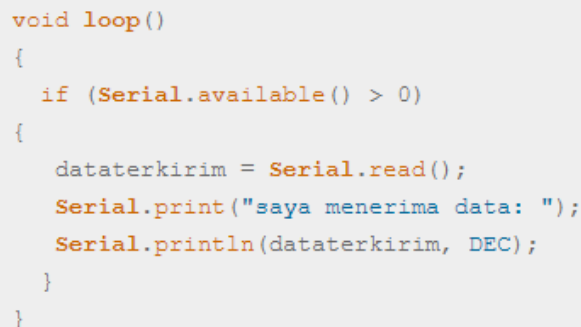


```
Serial.print(78)           // mencetak "78"
Serial.print(1.23456)      // mencetak "1.23"
Serial.print('N')         // mencetak "N"
Serial.print("Hello world.") // mencetak "Hello world."
Serial.print(78, BIN)      // mencetak "1001110"
Serial.print(78, OCT)      // mencetak "116"
Serial.print(78, DEC)      // mencetak "78"
Serial.print(78, HEX)      // mencetak "4E"
Serial.println(1.23456, 0)  // mencetak "1"
Serial.println(1.23456, 2) // mencetak "1.23"
Serial.println(1.23456, 4) // mencetak "1.2346"
```

Gambar 2.15 Serial print

Perintah “Serial.print” digunakan untuk menampilkan data ke serial monitor. Data yang ditampilkan dapat berupa karakter, bytes, atau angka. Berikut ini beberapa contoh perintah “serial.print()”. Terdapat 2 macam yaitu Serial.print dan Serial.println. perbedaannya adalah jika “Serial.print” akan menampilkan data di serial monitor dalam satu baris, tetapi perintah “Serial.println” akan menampilkan data di serial monitor pada baris baru.

6. Serial.read()



```
void loop()
{
  if (Serial.available() > 0)
  {
    dataterkirim = Serial.read();
    Serial.print("saya menerima data: ");
    Serial.println(dataterkirim, DEC);
  }
}
```

Gambar 2.16 Serial read

Perintah “Serial.read()” digunakan untuk membaca data dari serial port. Berikut contoh penulisan perintah “Serial.read()”

7. Serial.write()

```
void setup() {  
  Serial.begin(9600);  
}  
  
void loop() {  
  Serial.write(45);  
  int bytesSent = Serial.write("hello");  
}
```

Gambar 2.17 Serial write

Perintah “Serial.write” digunakan untuk membaca data biner dari serial port. Data ini dikirim dalam bentuk byte atau deretan data byte. Contoh penulisan perintah “Serial.write”.

Untuk mengatur komunikasi serial antar dua arduino dengan software IDE, digunakan source code seperti berikut.

```
#include <SoftwareSerial.h>  
  
SoftwareSerial serial(2,3);  
  
void setup() {  
  // put your setup code here, to run once:  
  serial.begin(9600);  
  serial.print("www.kendalielektro.com & www.ceceklawat.xyz");  
}  
  
void loop() {  
  // put your main code here, to run repeatedly.  
}
```

Gambar 2.18 Source code Arduino sebagai Transmitter

```

#include <SoftwareSerial.h>

SoftwareSerial serial(3,2);

String data;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  while (serial.available()>0){
    delay(10);
    char c = serial.read();
    data+=c;

  }if (data.length()>0){
    Serial.println(data);
    data="";
  }
}

```

Gambar 2.19 Source code Arduino sebagai Receiver

BAB III

KESIMPULAN

Dari pembahasan mengenai metrik USART (Universal Synchronous and Asynchronous Receiver Transmitter) diatas dapat ditarik kesimpulan sebagai berikut:

1. **USART** merupakan jenis perangkat keras komunikasi seperti register yang memungkinkan komputer untuk berkomunikasi dua arah mode sinkron dan asynchronous dengan perangkat yang terhubung secara serial
2. **USART** terdiri dari Register Data, Control dan Status (UDR, UCSRA, UCSRB, dan UCSRC)
3. Prinsip kerja **USART** pada transmisi sinkron pengirim akan mengirimkan clock / timing signal sehingga device penerima tahu kapan membaca bit data berikutnya. Transmisi asinkron mengijinkan pengirim tidak memberikan clock sinyal pada penerima, sebagai gantinya untuk memulai transmisi pengirim mengirimkan start bit pada tiap byte data yang dikirimkan dan diakhiri dengan stop bit
4. Untuk menggunakan komunikasi USART, dapat menggunakan Pin TX dan RX pada Arduino
5. Untuk mengatur komunikasi USART pada Arduino, dapat menggunakan Arduino IDE

DAFTAR PUSTAKA

- [1] Suyadi, 2012, Komunikasi Serial dan Port Serial (online),
<http://lutung.lib.ums.ac.id/arsip/publikasi/Bab-05-Komunikasi-Serial.pdf>,
diakses 9 Oktober 2018
- [2] <http://eprints.polsri.ac.id/3625/3/FILE%20III.pdf>, diakses 10 Oktober 2018
- [3] [http://edybillstephen.ilearning.me/2014/03/10/pengertian-serial-peripheral-interface-spi-dan-pengenalan-usart-atmega8535/\(online\)](http://edybillstephen.ilearning.me/2014/03/10/pengertian-serial-peripheral-interface-spi-dan-pengenalan-usart-atmega8535/(online)), diakses 10 Oktober 2018
- [4] <https://pccontrol.wordpress.com/2013/07/04/pengetahuan-dasar-pemrograman-usart-serial-komunikasi-avr-microcontroller> (online), diakses 19 Oktober 2018