Modul Praktikum Jaringan
Komunikasi PJK - 05
Protokol Pengiriman Data Asynchronous berbasis Mikrokontroler

Tujuan

1. Mengenal dan memahami protokol komunikasi data *asynchronous* berbasis mikrokontroler
2. Mengenal dan memahami prinsip kerja protokol komunikasi UART dan MODBUS pada mikrokontroler

Kompetensi Dasar

1. Rangkaian Listrik
2. Dasar Informatika
3. Elektronika Analog
4. Pengolahan Data
5. Elektronika Digital
6. Jaringan Komunikasi

Dasar Teori

UART (*Universal Asynchronous Receiver - Transmitter*)

UART adalah bagian perangkat keras komputer yang menerjemahkan bit-bit paralel data dan bit-bit serial. UART biasanya berupa sirkuit terintegrasi yang digunakan untuk komunikasi serial pada komputer atau *port serial* perangkat *peripheral*. Perangkat yang menggunakan protokol komunikasi UART dapat terhubung langsung pada perangkat lain yang juga mendukung protokol UART secara *wired* dan juga secara *wireless* dengan menggunakan modul RF433 [5], *bluetooth*, Wi-Fi, dll. Dalam pengiriman data,

baudrate antara pengirim dan penerima harus sama karena paket data dikirim tiap bit mengandalkan *baudrate* tersebut. Inilah salah satu keuntungan model *asynchronous* dalam pengiriman data karena dengan hanya satu kabel transmisi maka data dapat dikirimkan. UART merupakan antarmuka yang digunakan untuk komunikasi serial, seperti pada RS-232, RS-422, RS-485 [6].

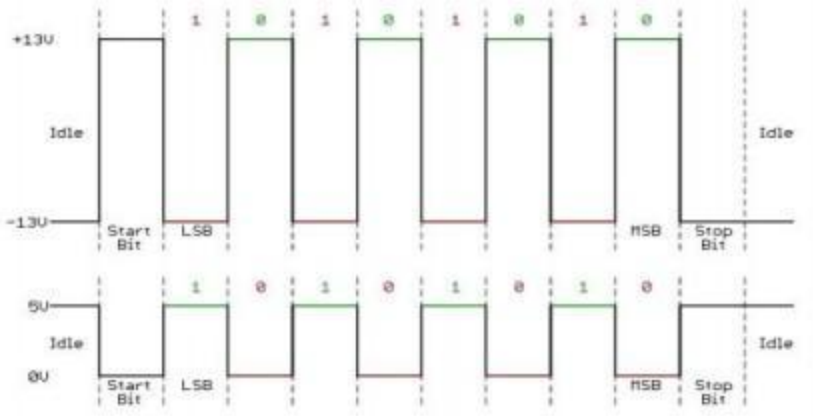
Komponen UART dapat dikonfigurasikan untuk *Full Duplex* (dua arah bersamaan), *Half Duplex* (dua arah bergantian), *Simplex* (hanya pengiriman atau penerimaan). Pada PSOC UART dapat dikonfigurasikan pada baudrate dari 110 sampai dengan 921600 bps dengan penyangga RX dan TX dari 4 sampai 65535[7].

Kebanyakan mikrokontroler saat ini telah membuat UART yang bisa digunakan untuk menerima dan mengirim data secara serial, metode ini biasanya disebut sebagai TTL serial (*transistor - transistor logic*). Komunikasi serial pada TTL selalu menggunakan batasan tegangan 0V dan Vcc yang biasanya sebesar 5V atau 3,3 V. Logika HIGH (‘1’) direpresentasikan oleh Vcc, sedangkan logika LOW (‘0’) direpresentasikan oleh 0V [8].

UART dapat dengan mudah dikonfigurasikan dengan memilih *baudrate, parity, bit data*, dan angka pada *start bits*. Pada konfigurasi RS232 biasanya disebut dengan “8N1” yang merupakan singkatan dari delapan *data bit*, tanpa *parity*, dan memiliki 1 *stop bit*, hal ini merupakan konfigurasi default pada komponen UART[7].

Port serial pada komputer mengikuti standar telekomunikasi RS-232 (*Recommended Standard* 232). Sinyal dari RS-232 sama dengan sinyal serial mikrokontroler apabila sinyal tersebut ditransmisikan pada satu bit dalam satu waktu, dengan *baudrate* tertentu dan dengan atau tanpa *parity* maupun *stop bits*. Pada Standar RS-232 maka *logic* HIGH (‘1’) direpresentasikan oleh tegangan negatif (-3V sampai – 25 V) dan logic LOW direpresentasikan dengan tegangan positif (3V sampai 25V). Kebanyakan PCs (*personal communication system*) memiliki sinyal dari -13 V sampai 13 V [8].

Karena mempunyai rentang tegangan yang lebar maka sinyal RS232 membuat data yang dikirim tahan terhadap derau, gangguan, dan degradasi, sehingga dapat dikatakan bahwa sinyal RS-232 dapat melakukan komunikasi pada jarak yang jauh dibanding dengan serial TTL [8].



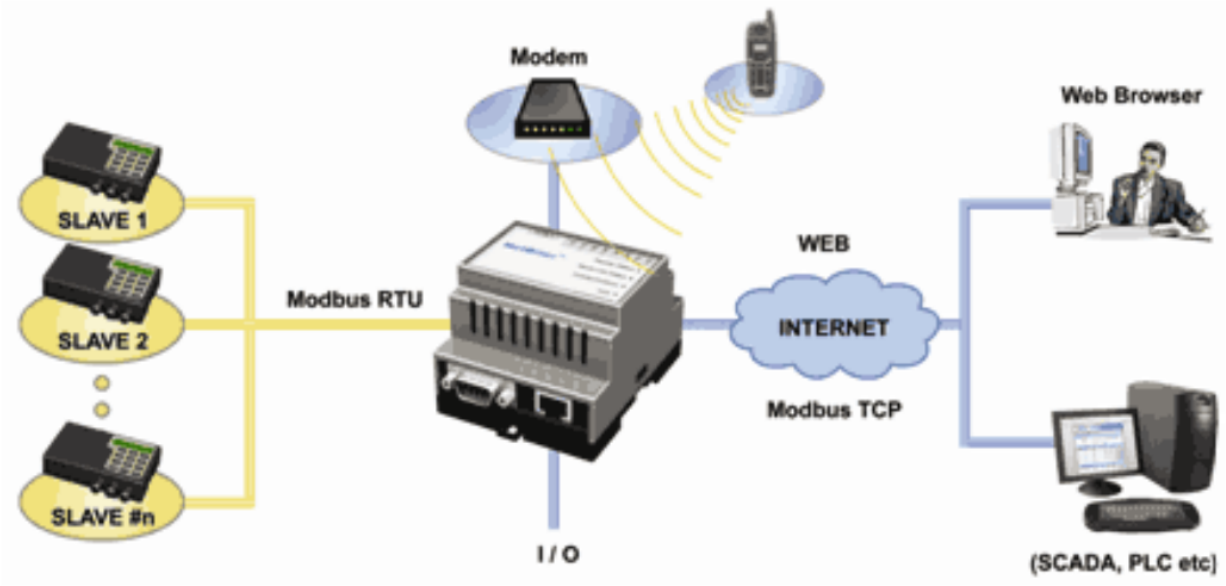
Gambar 1.1. Pengiriman sinyal TTL dan RS-232.

MODBUS

Pendahuluan

Modbus adalah protokol komunikasi serial yang dipublikasikan oleh Modicon pada tahun 1979 untuk diaplikasikan ke dalam programmable logic controller (PLC). Modbus terletak pada level 7 dari model OSI (OSI Layer). Modbus sudah menjadi standar protokol yang umum digunakan untuk menghubungkan peralatan elektronik industri. Terdapat beberapa alasan protokol ini banyak digunakan, yaitu:

1. Modbus dipublikasikan secara terbuka dan bebas royalti.
2. Mudah digunakan dan dipelihara
3. Memindahkan data bit atau word tanpa banyak membatasi vendor.



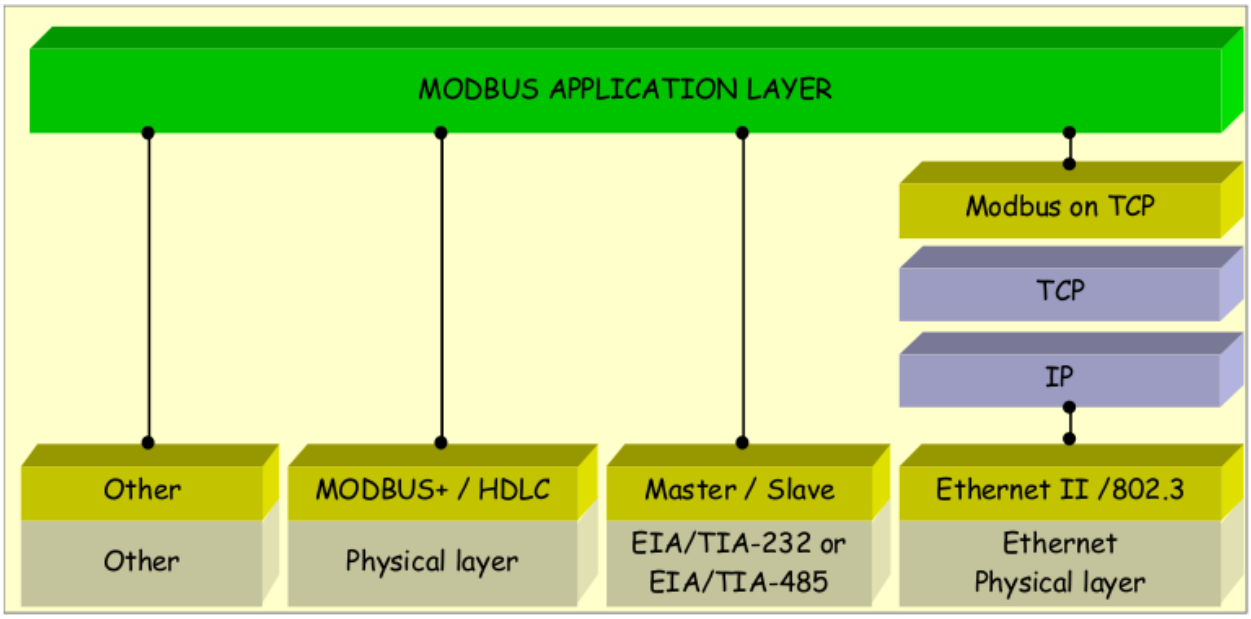
Gambar 1.2. Ilustrasi Protokol Komunikasi Modbus

Modbus digunakan untuk komunikasi antar banyak perangkat dalam satu jaringan. Modbus banyak digunakan untuk menghubungkan komputer pemantau dengan remote terminal unit (RTU) pada sistem supervisory control and data aquisition (SCADA).

Jenis-jenis Modbus

Terdapat beberapa jenis versi Modbus, antara lain:

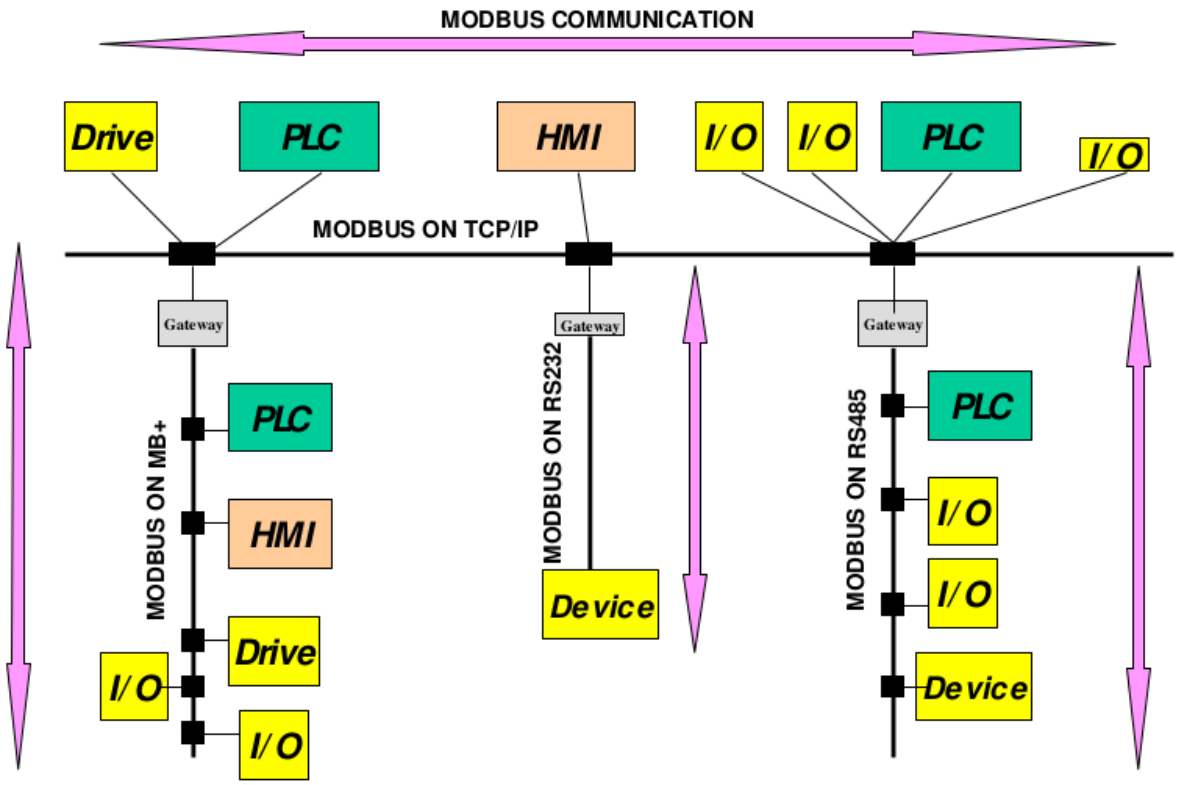
1. Modbus RTU: Varian Modbus yang ringkas dan digunakan pada komunikasi serial. Format RTU dilengkapi dengan mekanisme cyclic redundancy error (CRC) untuk memastikan keandalan data. Modbus RTU merupakan implementasi protokol Modbus yang paling umum digunakan. Setiap frame data dipisahkan dengan periode idle (silent).
2. Modbus ASCII: Digunakan pada komunikasi serial dengan memanfaatkan karakter ASCII. Format ASCII menggunakan mekanisme longitudinal redundancy check (LRC). Setiap data frame data Modbus ASCII diawali dengan titik dua (“:”) dan baris baru yang mengikuti (CR/LF).
3. Modbus TCP/IP atau Modbus TCP merupakan Modbus yang digunakan pada jaringan TCP/IP. Komunitas internet dapat mengakses Modbus di Port sistem 502 pada TCP/IP.



Gambar 1.3. Lapisan Penggunaan Modbus

Arsitektur Jaringan Modbus

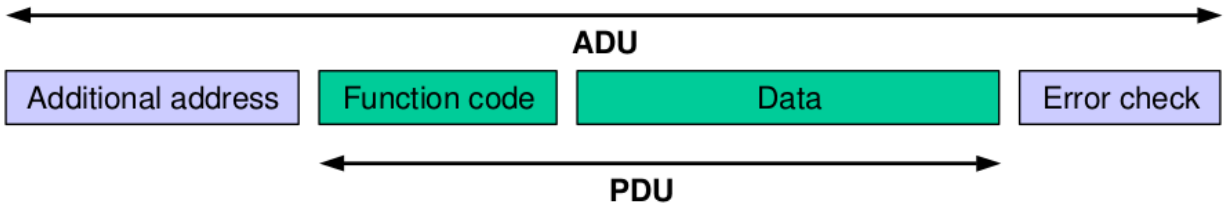
Setiap jenis perangkat, baik itu PLC, HMI, Control Panel, Driver, Motion Control, Perangkat I/O, dapat menggunakan protokol Modbus untuk memulai sebuah komunikasi jarak jauh. Komunikasi yang sama dapat dilakukan juga pada jalur serial sama seperti jaringan ethernet TCP/IP. Perangkat Gateway memungkinkan komunikasi antar beberapa tipe bus atau jaringan menggunakan protokol Modbus.



Gambar 1.4. Arsitektur Komunikasi Modbus

Gambaran Protokol

Modbus adalah protokol *request* / *reply* dan menawarkan layanan yang ditentukan oleh kode fungsi. Kode fungsi Modbus merupakan unsur *request* / *reply* PDU Modbus. Protokol Modbus mendefinisikan sebuah protokol data unit (PDU) independen pada lapisan komunikasi. Pemetaan protkol Modbus pada bus yang spesifik atau jaringan dapat mengenalkan beberapa penambahan isian data pada sebuah Application Data Unit (ADU).



Gambar 1.5. Gambaran umum protokol Modbus

Application Data Unit (ADU) dari sebuah Modbus protokol dibangun oleh klien yang menginisiasi sebuah transaksi data. Fungsi tersebut mengindikasikan ke server jenis aksi yang akan dilakukan. Kemudian protokol Modbus membangun format yang telah diminta oleh klien.

Sebagian besar peralatan Modbus menggunakan port serial RS-485. Konsep dasar komunikasi Modbus terdiri master dan slave. Peralatan yang bertindak sebagai slave akan terus idle kecuali mendapat perintah dari master. Setiap Peralatan yang dihubungkan (slave) harus memiliki alamat unik. Sebuah perintah Modbus dilengkapi dengan alamat tujuan perintah tersebut. Hanya alamat tujuan yang akan memproses perintah, meskipun peralatan yang lain mungkin menerima perintah tersebut. Setiap perintah modbus memiliki informasi pemeriksaan kesalahan untuk memastikan data diterima tanpa kerusakan. Perintah dasar Modbus RTU dapat memerintahkan peralatan untuk mengubah nilai registernya, mengendalikan dan membaca port I/O, serta memerintahkan peralatan untuk mengirimkan kembali nilai yang ada pada registernya.

Cara kerja Modbus

Modbus serial adalah komunikasi serial yang menggunakan protokol Modbus. Pada komunikasi serial, pengiriman data besarnya adalah 8 byte atau 8 frame data dalam 1x pengiriman.

[byte0] [byte1] [byte2] [byte3] [byte4] [byte5] [byte6] [byte7]

Besarnya nilai data dalam 1 framenya adalah 2^8 = 256 (0-255). maka nilai terbesarnya adalah 256. jika kita mengirimkan ASCII misalkan huruf “A” maka nilai yang digunakan sesuai ASCII table adalah 65. Pada Modbus frame-frame tersebut diisi dengan kesepakatan tertentu. Modbus frame untuk request dari master ke slave adalah sebagai berikut,

[address RTU] [Function Code] [Reg] [Reg] [Lenght][Lenght] [CRC1] [CRC2]

Keterangan:

Address RTU / Slave: Address dari setiap RTU, jika alamat RTU yang direquest masuk ke RTU sesuai, maka RTU tersebut akan meresponnya dengan memberikan pesan balasan. Byte pertama sebagai alamat *slave* terdiri dari 1 byte (0~255). Namun alamat slave yang dapat diakses hanya 1 hingga 247. Alamat 0 ditujukan untuk perangkat master.

Function Code: Byte kedua berupa function code, yaitu perintah fungsi akses data dari master yang harus dilakukan oleh *slave*. Berikut ini adalah jenis-jenis fungsi yang dapat digunakan yaitu,

- 01 *Read Coil Status*
- 02 *Read Input Status*
- 03 *Read Holding Registers*

- 04 *Read Input Registers*
- 05 *Force Single Coil*
- 06 *Force Single Register*
- 07 Read Exception Status
- 08 Diagnostics
- 09 Program 484
- 10 Poll 484
- 11 Fetch Communication Event Counter
- 12 Fetch Communication Event Log
- 13 Program Controller
- 14 Poll Controller
- 15 *Force Multiple Coils*
- 16 *Preset Multiple Registers*
- 17 Report Slave ID
- 18 Program 884 / M84
- 19 Reset Comm. Link
- 20 Read General Reference
- 21 Write General Reference
- 22 Mask Write 4X Register
- 23 Read / Write 4X Registers
- 24 Read FIFO Queue.

Tabel 1.1. Register pada Modbus RTU

Data	Read 1 Data	Write 1 Data	Write Multiple Data	No. Awal Register
Coil	FC01	FC05	FC15	00001
Input Digital	FC02			10001
Input Register	FC04			30001
Holding Register	FC03			40001

Reg: merupakan register yang ingin diambil nilainya. Frame ini menggunakan 2 byte data, maka dari itu nilai register sampai 256*256 namun dibatasi dengan jumlah register yang tersedia.

Length: merupakan jumlah register yang ingin direquest. Ini juga menggunakan 2 byte frame seperti register.

Error Check (CRC1 dan CRC2): CRC adalah Cyclic Redundancy Check yaitu sebuah metode untuk pengecekan error. CRC ini menggunakan 2 byte frame juga, untuk menghitung dan mengetahui lebih jelas dapat dilihat di CRC. Cara kerja CRC secara sederhana adalah menggunakan perhitungan matematika terhadap sebuah bilangan yang disebut dengan Checksum, yang dibuat berdasarkan total bit yang hendak ditransmisikan atau yang hendak disimpan.

Penyimpanan data pada Modbus

Pada protokol Modbus terdapat 4 jenis akses data dengan panjang masing-masing 16 bit.

Tabel 1.2. *Function Code Modbus*

Nama Register	Tipe Objek	Akses	Keterangan
Coils	Single Bit	Read-Write	Master bisa membaca maupun menulis data coil
Discrete Input	Single Bit	Read Only	Data hanya bisa diubah oleh slave
Input Register	16 bit Word	Read Only	Data hanya bisa diubah oleh slave
Holding Register	16-bit Word	Read-Write	Master bisa membaca dan menulis register

a) Coil: Register ini merupakan register untuk menyimpan nilai diskrit on atau off. Panjang data setiap register adalah 16 bit. Dimana untuk data yang digunakan untuk mengaktifkan coil (ON) bernilai 0xFF00. Sedangkan untuk menonaktifkan coil (OFF) bernilai 0x0000. Data ini dapat disimpan di register 00000 sampai 09999.

b) Input Relay

   Kebalikan dengan coil, input relay digunakan untuk mengetahui status relay apakah sedang dalam kondisi ON atau OFF. Input relay bersifat read only bagi master dan hanya bisa dirubah oleh slave yang bersangkutan. Data tersebut disimpan di register 10001 sampai 19999.

c) *Input Register*

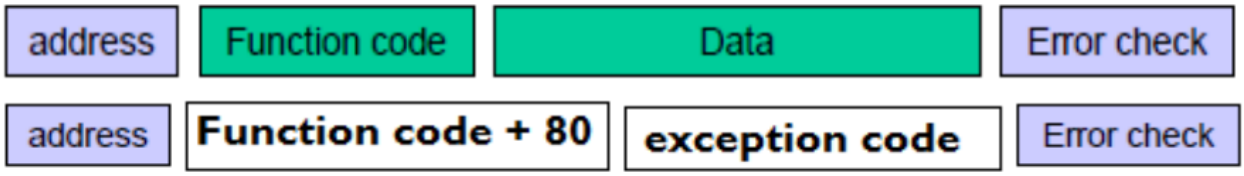
Input register digunakan untuk menyimpan data analog dengan range nilai 0 hingga 65535 . Input register bersifat *read only* bagi master. Data ini disimpan di register 30001 sampai 39999

d) Holding Register

Holding register digunakan untuk menyimpan nilai dengan range 0 – 65535. Register ini mempunyai alamat register 40001 sampai 49999.

e) Respon Modbus Exception

Respon exception adalah respon dari slave ketika terjadi keadaan tidak normal / *error*. *Slave* menerima *query*, tetapi *slave* tidak dapat menangani perintah tersebut. *Slave* akan mengirimkan sebuah respon exception. Frame respon jika terjadi kesalahan berbedan dengan frame dalam keadaan normal.



Gambar 1.6. Perbedaan frame Modbus normal dan saat terjadi exception

Bila terjadi kesalahan pada pengiriman data, maka slave akan mengirimkan informasi kepada master dengan pesan *exception frame*. Berikut ini adalah daftar exception code yang terjadi bila terdapat kesalahan dalam pengiriman data:

Tabel 1.3. Exception Code

Exception Code	Nama Exception	Keterangan
01	*ILLEGAL FUNCTION*	Function code salah
02	ILLEGAL DATA ADDRESS	Alamat register salah atau tidak tersedia.
03	ILLEGAL DATA VALUE	Mengandung nilai data yang tidak diizinkan slave.
04	SLAVE DEVICE FAILURE	Slave gagal melaksanakan perintah master
05	ACKNOWLEDGMENT	Pemberitahuan ke master bahwa pelaksanaan perintah akan memakan waktu yang lama, sehingga mengalami time out.
06	SLAVE DEVICE BUSY	Slave sedang sibuk

Modbus Message Respon

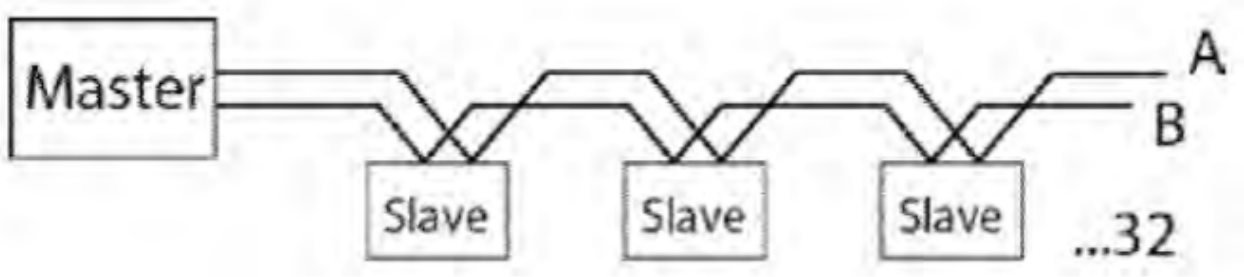
Pesan respon dari slave ke RTU memiliki sedikit perbedaan dengan request. Untuk pesan Modbus dari slave memiliki frame yang menyesuaikan jumlah data yang dikirimkan dimana

setiap 1 data menggunaka 2 byte frame. susunan nya adalah:

[RTU Address] [Function Code] [2*panjang data] [data] [data][data…] [data…] [CRC1] [CRC2]

Komunikasi RS-485

RS485 adalah adalah teknik komunikasi data serial yang dikembangkan di tahun 1983 di mana teknik ini, komunikasi data dapat dilakukan pada jarak yang cukup jauh yaitu 1,2 km. Komunikasi RS485 dapat digunakan untuk komunikasi multidrop yaitu berhubungan secara one to many dengan jarak yang jauh. Teknik ini dapat digunakan untuk menghubungkan 32 unit beban sekaligus hanya dengan menggunakan dua buah kabel saja tanpa memerlukan referensi ground yang sama antar unit yang satu dengan unit lainnya.



Gambar 1.7. Topologi RS-485

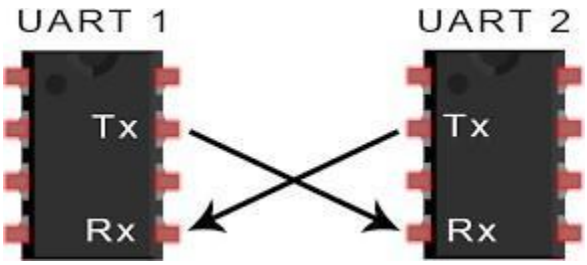
Bus RS485 adalah mode transmisi balanced differential. Bus ini hanya mempunyai dua sinyal, A dan B dengan perbedaan tegangan antara keduanya. Karena line A sebagai referensi terhadap B maka sinyal akan high bila mendapat input low demikian pula sebaliknya. Pada komunikasi RS485, semua peralatan elektronik berada pada posisi penerima hingga salah satu memerlukan untuk mengirimkan data, maka peralatan tersebut akan berpindah ke mode pengirim, mengirimkan data dan kembali ke mode penerima. Setiap kali peralatan elektronik tersebut hendak mengirimkan data, maka terlebih dahulu harus diperiksa, apakah jalur yang akan digunakan sebagai media pengiriman data tersebut tidak sibuk. Apabila jalur masih sibuk, maka peralatan tersebut harus menunggu hingga jalur sepi.

Agar data yang dikirimkan hanya sampai ke peralatan elektronik yang dituju, misalkan ke salah satu Slave, maka terlebih dahulu pengiriman tersebut diawali dengan Slave ID dan dilanjutkan dengan data yang dikirimkan. Peralatan elektronik yang lain akan menerima data tersebut, namun bila data yang diterima tidak mempunyai ID yang sama dengan Slave ID yang dikirimkan, maka peralatan tersebut harus menolak atau mengabaikan data tersebut. Namun bila Slave ID yang dikirimkan sesuai dengan ID dari peralatan elektronik yang menerima, maka data selanjutnya akan diambil untuk diproses lebih lanjut.

Langkah Percobaan

UART

Buatlah rangkaian yang menggunakan Arduino dan PSoC seperti pada gambar blok diagram dibawah dengan menghubungkan TX dan RX pada perangkat 1 dan 2 secara bersilangan.



Percobaan 1: Komunikasi UART antar Arduino

a) Unduh Arduino IDE versi 1.8.18 pada tautan berikut ini, Previous IDE Releases | Arduino.

b) Buka Arduino IDE

c) Buat program untuk pengiriman data protokol komunikasi UART menggunakan Arduino 1 seperti pada Gambar 2.2.

```
void setup() {
  serial.Begin(19200);

}

void loop() {
  serial.print("C");
}
delay(50000);
```

Gambar 2.2. Contoh kode program untuk pengiriman data UART menggunakan Arduino

d) Buat program untuk penerimaan data protokol komunikasi UART menggunakan Arduino

2 seperti Gambar 2.3.

```
char data_rcvd;
void setup(){
  Serial.Begin(9600);
}
void loop(){
  if(serial.available()){
    data_received = Serial.Read();
    Serial.print(Data_received);
  }
}
```

Gambar 2.3. Contoh kode program penerimaan data UART menggunakan Arduino

e) Verify program untuk memastikan program yang dibuat sudah tepat atau belum. Jika sudah, tekan *build* untuk memasukkan program yang dibuat ke masing-masing Arduino.

f) Jalankan percobaan untuk 2 karakter, karakter yang digunakan sesuai kesepakatan dengan masing-masing asisten praktikum.

g) Buka Arduino IDE pada laptop yang tersambung dengan Arduino 2, lalu buka serial monitor.

h) Pastikan baudrate pada kode program Arduino 2 sama dengan baudrate serial monitornya.

i) Variasikan Baudrate seperti pada tabel percobaan.

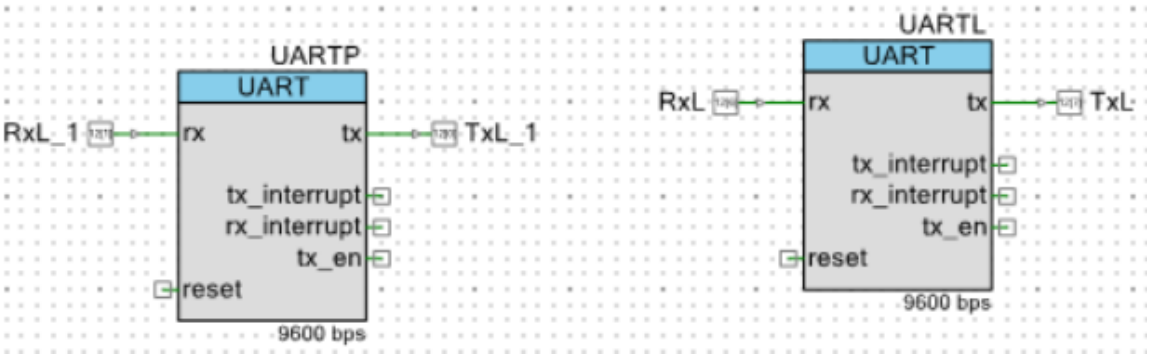
j) Catat hasil yang didapatkan pada tabel percobaan.

Percobaan 2: Komunikasi UART antar PSoC

a) Buka PSoC Creator

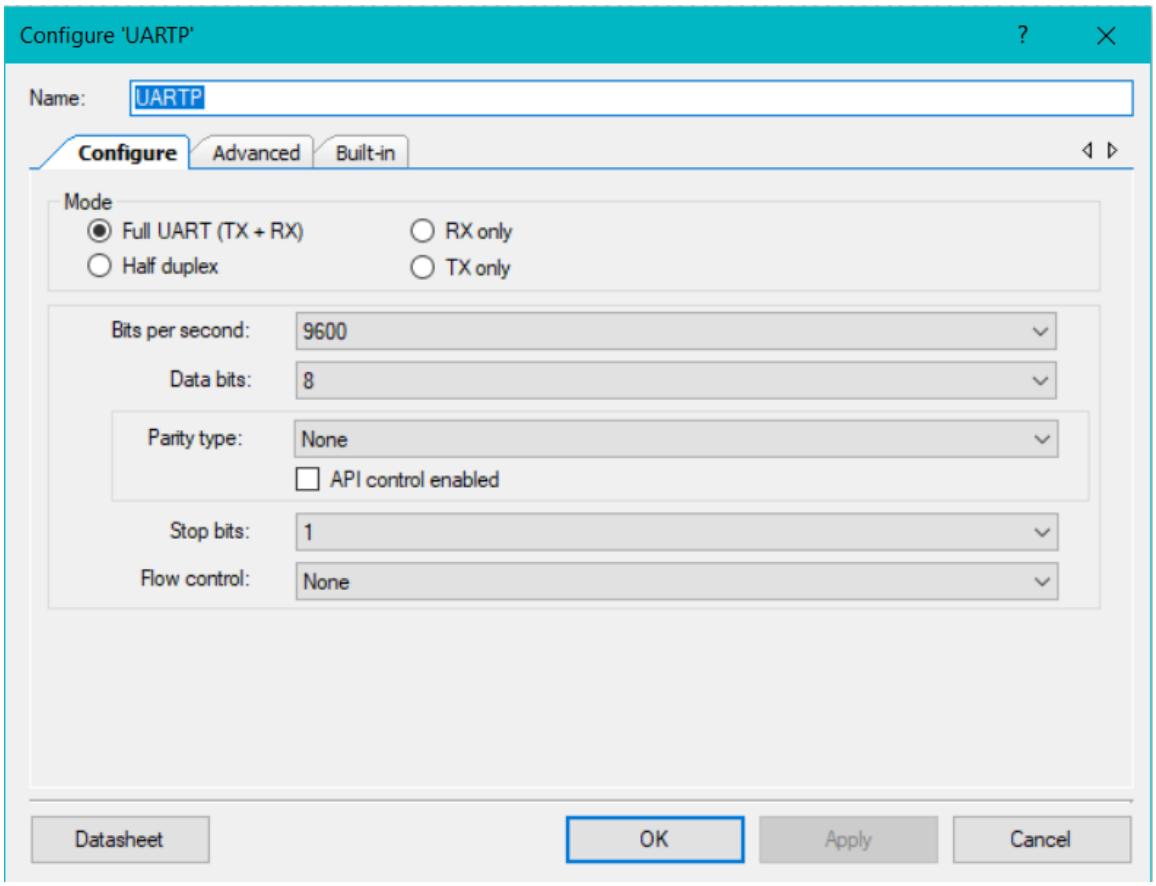
b) Masuk ke bagian TopDesign.cysch.

c) Masukkan komponen UART yang didapatkan dari Component Catalog, seperti



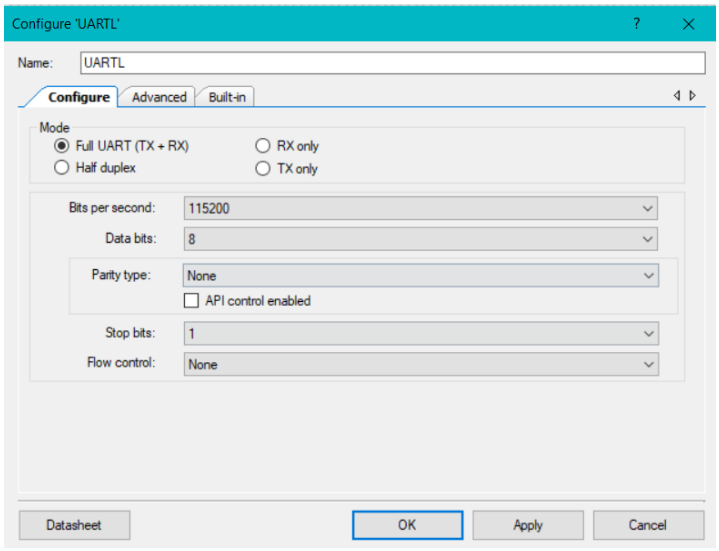
Gambar 2.4. Blok Komponen komunikasi UART antar PSoC

d) Atur konfigurasi blok UART PSoC 1 seperti Gambar 2.4.



Gambar 2.5. Pengaturan blok UART PSoC 1

e) Atur konfigurasi blok UART PSoC 2 seperti Gambar 2.6.



Gambar 2.6. Pengaturan blok UART PSoC 2

f) Buat kode program seperti Gambar 2.7.

```
#include "project.h"
char result;
char result2;

int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */
    UARTP_Start();
    UARTL_Start();
    /* Place your initialization/startup code here (e.g. MyInst_Start()) */

    for(;;)
    {
        result = "Jarkom Hore";
        if (result = "Jarkom Asyik"){
            UARTP_PutChar(result);
        }
        result2 = UARTP_GetChar();
        if (result2){
            UARTL_PutChar(result2);
        }
    }
}

/* [] END OF FILE */
```

Gambar 2.7. Contoh kode program PSoC protokol komunikasi UART

g) Atur ports P12[0] sebagai pin Tx dan P12[1] sebagai Rx

h) Konfigurasi port Rx dan Tx yang tersambung dengan laptop adalah P12[6] dan P12[7].

i) Variasikan baudrate PSoC 1 dan PSoC 2 seperti tabel percobaan.

j) Buka Arduino IDE pada laptop yang tersambung dengan PSoC 2, lalu buka serial monitor

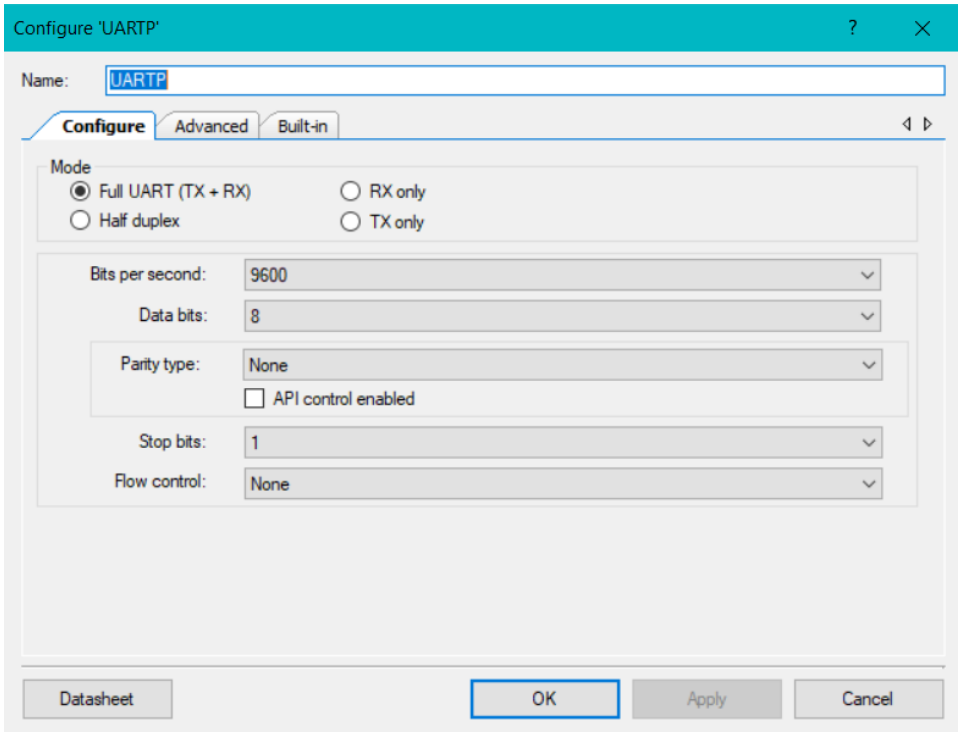
k) Pastikan *baudrate* pada PSoC 2 sama dengan *baudrate* serial monitornya.
l) Amati hasil yang didapatkan pada PSoC 2 yang terhubung pada komputer 2 (dua)

Percobaan 3: Data Frame protokol komunikasi UART menggunakan Arduino

a) Buka Arduino IDE
b) Buat program untuk pengiriman data protokol komunikasi UART menggunakan Arduino seperti pada Gambar 2.2.
c) Sambungkan pin Tx Arduino pada osilloskop.
d) Verify program untuk memastikan program yang dibuat sudah tepat atau belum. Jika sudah, tekan *build* untuk memasukkan program yang dibuat ke Arduino.
e) Jalankan percobaan untuk 2 karakter, karakter yang digunakan sesuai kesepakatan dengan masing-masing asisten praktikum.
f) Variasikan stop-bit dan parity-bit seperti pada tabel percobaan
g) Amati data frame yang tertampil di osiloskop.

Percobaan 4: Data Frame protokol komunikasi UART menggunakan PSoC

a) Buka PSoC Creator
b) Masuk ke bagian TopDesign.cysch.
c) Masukkan komponen UART yang didapat dari Component Catalog.
d) Atur konfigurasi blok UART seperti Gambar 2.8.



Gambar 2.8. Konfigurasi Blok UART di PSoC

e) Buat kode program pada main.c PSoC Creator seperti Gambar 2.9.

```
#include "project.h"
char result;
char result2;

int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */
    UARTA_Start();
    UARTB_Start();
    /* Place your initialization/startup code here (e.g. MyInst_Start()) */

    for(;;)
    {
        result = UARTL_GetChar();
        if (result){
            UARTP_PutChar(result);
        }
        result2 = UARTP_GetChar();
        if (result2){
            UARTL_PutChar(result2);
        }
    }
}

/* [] END OF FILE */
```

Gambar 2.9. Contoh kode program PSoC untuk komunikasi UART

f) Konfigurasi ports PSoC yang terhubung komputer adalah P12[6] sebagai Rx dan P12[7] sebagai Tx.

g) *Clean and build* lalu unggah program ke masing-masing modul PSoC

h) Variasikan *stop-bit* dan *parity-bit* sesuai tabel percobaan 3 (tiga).

i) Amati data frame menggunakan osiloskop dengan menghubungkan pin Tx PSoC dengan probe osiloskop.

Percobaan 5: Protokol Komunikasi UART antara Laptop, Arduino, dan PSoC

a) Buka ENV program Python pada komputer Anda.

b) Buat kode program seperti pada Gambar 2.10.

```
from time import sleep
import serial

baudrate = 9600
COM = 'COM14'


com = serial.Serial(baudrate, COM)
data_kirim = "A"
for i in range(900):
    com.write(data_kirim.encode("utf-8"))
    print(i)
    sleep(1)
com.close()
```

Gambar 2.10. Contoh kode program UART untuk mengirimkan data menggunakan Python

c) Siapkan Mikrokontroler Arduino

d) Buka Arduino IDE

e) Buat kode program Arduino seperti pada Gambar 2.11.

```
String data_terima, data_mentah;
void setup() {
  Serial.begin(9600);
}

void loop() {
  while (Serial.available()
  {
    data_mentah = Serial.read();
    data_terima = Serial.readString();
  }
  Serial.print("Data mentah: ");
  Serial.println(data_mentah);
  Serial.print("Data yang diterima: ");
  Serial.println(data_terima);
 }
```

Gambar 2.11. Contoh kode program UART untuk Arduino Receiver

f) *Verify* dan *build* kode program Gambar 2.11 pada Arduino.

g) Siapkan PSoC 5LP

h) Buka PSoC Creator

i) Masuk ke bagian TopDesign.cysch.

j) Masukkan komponen UART yang didapat dari Component Catalog.

k) Atur konfigurasi blok UART seperti Gambar 2.6.

l) Buat kode program pada main.c PSoC Creator seperti Gambar 2.12.

```
#include "project.h"
char result;
char result2;

int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */
    UART1_Start();
    UART2_Start();
    /* Place your initialization/startup code here (e.g. MyInst_Start()) */

    for(;;)
    {
        result = UART3_GetChar();
        if (result){
            UART4_PutChar(result);
        }
        result2 = UART3_GetChar();
        if (result3){
            UART4_PutChar(result2);
        }
    }
}

/* [] END OF FILE */
```

Gambar 2.12. Kode Program UART Full Duplex pada PSoC

m) Konfigurasi port blok UART PSoC yang tersambung dengan Arduino adalah P12[0] sebagai pin Tx dan P12[1] sebagai Rx

n) Konfigurasi port blok UART PSoC yang tersambung dengan laptop adalah P12[6] sebagai Rx dan P12[7] sebagai Tx.

o) Buat kode program python seperti Gambar 2.13 pada laptop 2

```python
import serial
import time

con = serial.Serial("COM14", "9600") # Sesuaikan port dengan port yang tersambung dengan mikrokontroler
FileBaca = "File Hasil Baca Serial.txt"

waktu_awal = time.time()

def BacaSerial():
    while(con.inWaiting()==0):
        pass
    data_awal = str(con.readline(), "utf-8")
    open(FileBaca,"a").write(data_awal)
    open(FileBaca,"a").close()
    data_jadi = data_awal.splitlines()[0].split(";").rsplit("\r\n")
    print(data_jadi)

waktu_akhir = time.time()
print(f"waktu untuk pengiriman data: {waktu_akhir - waktu_awal}")
BacaSerial()
```

Gambar 2.13. Contoh kode program Python untuk baca data UART

p) Sambungkan PSoC dengan Laptop 2.

q) Jalankan program Gambar 2.10 dan Gambar 2.13 pada terminal laptop 1 dan 2.

r) Variasikan data yang dikirim pada Gambar 2.10 sesuai tabel percobaan.

s) Amati dan catat data yang ditampilkan dan waktu pengiriman data pada terminal laptop 2.

MODBUS

Percobaan 6: Interface Modbus

a) Buka Arduino IDE

b) Tambahkan *library* ModbusRtu dengan menambahkan berkas zip dari *repository* berikut: smarmengol/Modbus-Master-Slave-for-Arduino: Modbus Master-Slave library for Arduino (github.com)

c) Buat kode program Modbus *Slave* dan unggah pada Arduino

```
#include <ModbusRtu.h>
#define TXEN  3

uint16_t au16data[8] = {
  0, 25, 50, 11, 33, 45, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
int Ledstatus = 0;
int ledPin = 13;

Modbus slave(1,serial,TXEN);

void setup() {
  serial.begin(19200);
  pinMode(ledpin,OUTPUT);
  slave.start();
}

void loop() {
  slave.poll(au16data,16);
  if(ledstatus != au16data[0]) {
    ledstatus = au16data[0];
    digitalWrite(ledpin,ledstatus);
  }
}
```

d) Buat kode program Modbus *Master* pada Laptop

```
import Serial
import minimalmodbus

instrument = minimalmodbus.Instrument('COM', 1)
Instrument.serial.baudrate = 19200

while True:
    ledState = input("Set LED state (0/1): ")
    registerselect = input("Select Register (0-15): ")
    instrument.write_register(0, int(ledState), 0)
    readRegister = instrument.read_register(registerSelect, 0)
    print("readRegister")
```

e) Hubungkan Arduino dengan modul MAX485 dengan RS485-to-USB dengan ketentuan:

Arduino	MAX485
5V	VCC
GND	GND
Digital 3	RE
Digital 3	DE
Digital 0	R0
Digital 1	DI

f) Hubungkan pin A dan B pada modul MAX485 dengan pin A dan B pada RS485-to-USB
g) Jalankan program modbus *Master*
h) Catat dan variasikan percobaan sesuai dengan tabel percobaan

Tabel Percobaan

Percobaan 1 dan 2: Protokol Komunikasi Antar Arduino dan PSoC

Tabel 3.1. Hasil Percobaan Variasi *Baudrate*

Mikrokontroler	Karakter yang dikirim	Baudrate Pengirim	Baudrate Penerima	Karakter yang diterima
Arduino	(Pertama)	4800	4800	
		115200	115200	
		115200	4800	
		9600	19200	
PSoC	(Kedua)	4800	4800	
		19200	19200	
		115200	4800	
		9600	19200	
Kesimpulan				

. **Percobaan 3 : Data Frame Protokol Komunikasi UART menggunakan Arduino**

Tabel 3.2. Percobaan Data Frame menggunakan Arduino

Mikrokontroler	**Karakter yang dikirim**	**Stop-bit**		**Parity**			**Kombinasi biner 8 bit**	**Nomor ASCII**
		1	**2**	**Odd**	**Even**	**None**		
Arduino	(Pertama)	v				v		
		v		v				
		v			v			
			v			v		
	(Kedua)	v				v		
		v		v				
		v			v			
			v			v		
Kesimpulan								

Percobaan 4: Data Frame Protokol Komunikasi UART menggunakan PSoC

Tabel 3.3. Percobaan Data Frame UART menggunakan PSoC

Mikrokontroler	**Karakter yang dikirim**	**Stop-bit**		**Parity**			**Kombinasi Biner 8 bit**	**No ASCII**
		1	**2**	**Odd**	**Even**	**None**		
PSoC	(Pertama)	v				v		
		v		v				
		v			v			
			v			v		
	(Kedua)	v				v		
		v		v				
		v			v			
			v			v		
Kesimpulan								

Jangan lupa foto data frame hasil percobaan Tabel 3.2 dan Tabel 3.3.

Percobaan 5: Komunikasi UART antara Laptop, Arduino dan PSoC

Tabel 3.4. Percobaan Komunikasi UART antar Board

Mikrokontroler Transmitter	Karakter yang dikirim	Baudrate	Mikrokontroler Receiver	Karakter yang diterima
Arduino	A		PSoC	
	3.65			
PSoC	C		Arduino	
	&#;			
Kesimpulan				

Percobaan 6: Interface Modbus

Tabel 3.5. Percobaan uji coba Modbus

Arduino		**Laptop**			**Panjang Kabel A&B (m)**	**Hasil**
Slave Address	**Register Limit**	**Slave Address**	**Register Read**	**LED State**		
1	**16**	**1**	**5**	**1**	**1**	
1	**16**	**5**	**4**	**1**	**1**	
1	**4**	**1**	**7**	**1**	**1**	
1	**4**	**1**	**2**	**0**	**1**	
1	**16**	**1**	**7**	**0**	**1**	
1	**16**	**1**	**3**	**1**	**2**	
…	**…**	**…**	**…**	**…**	**…**	