CPS Design Lab 1: Communication between PC and Arduino processor

Semester 1-2022

Objectives

- To be familiar with the serial communications
- To learn how to communicate with an Arduino using Python.

UART, SPI and I^2C,

Why aren’t these good enough for everything?

- UART

  Slow. No shared bus.

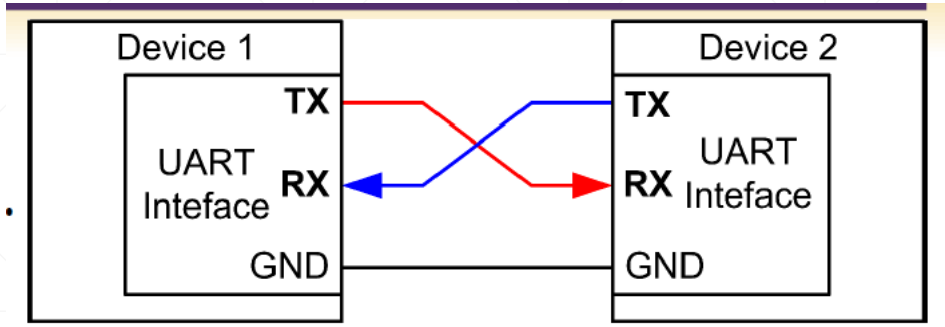
- I^2C

  Slow. Master-initiated communication.

- SPI

  Master-initiated communication. Lots of pins.

Universal Asynchronous Receiver and Transmitter (UART)



- Universal

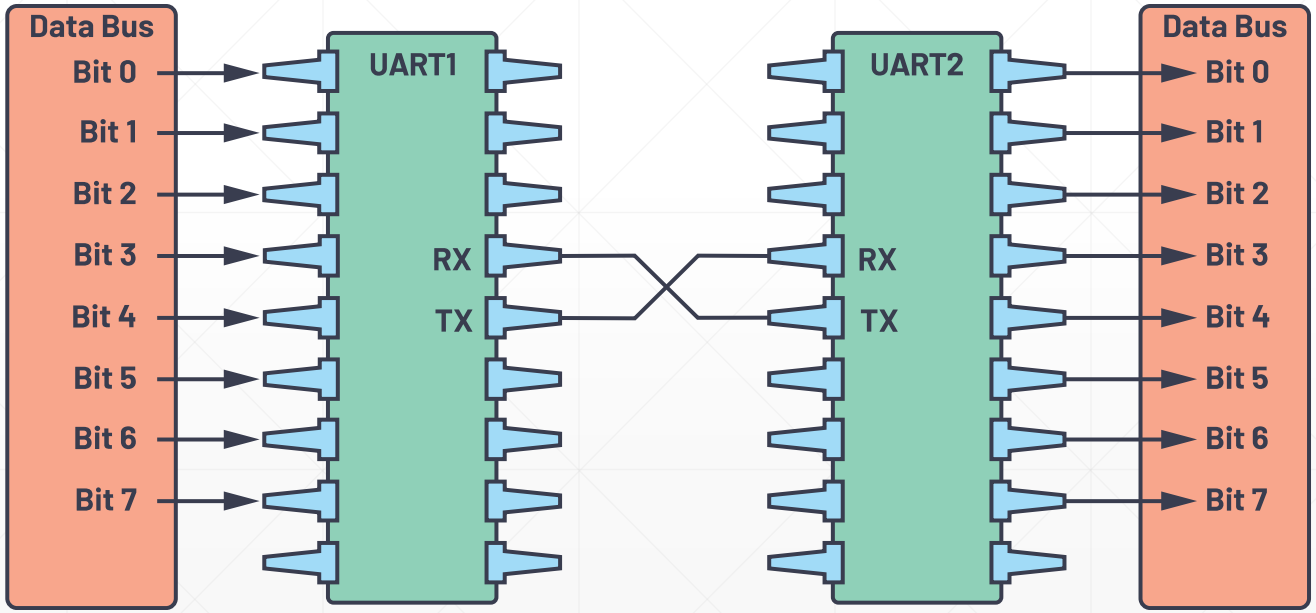
  Programmable format, speed, etc.
- Asynchronous

  Sender provides no clock signal to receivers
- Half Duplex
- Any node can initiate communication
- Two lanes are independent of each other
- Not a communication protocol, but a physical circuit in a μ controller or a stand-alone IC.
- Sender and receiver use the same transmission speed (10% clock shift/difference is tolerated)

Tow UARTs communicate directly.

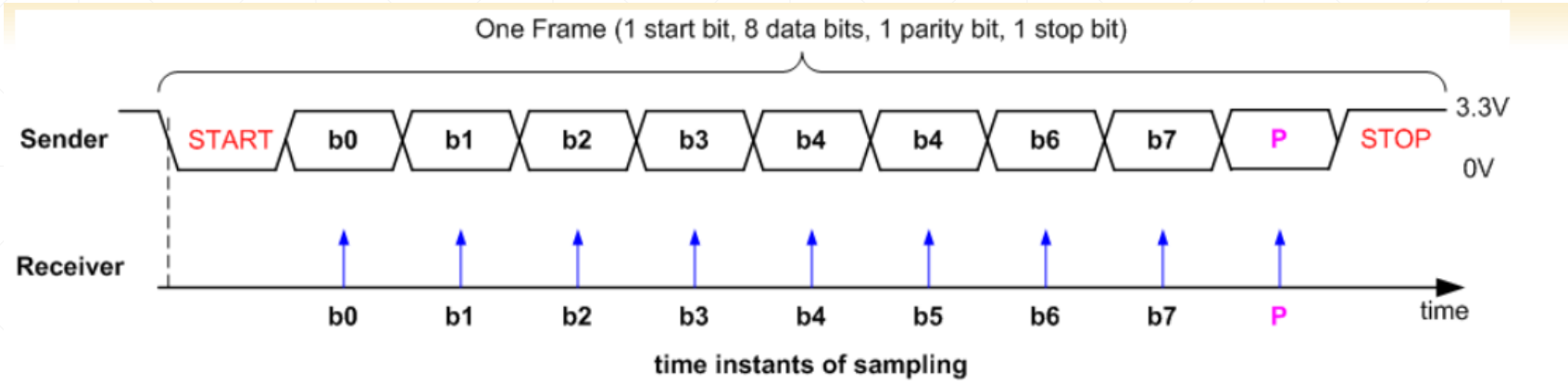
- The transmitting UART converts parallel data from a controlling device like a CPU into serial form, transmits it in serial to the receiving UART, which then converts the serial data back into parallel data for the receiving device.

•Transmitter (Tx)
•Receiver (Rx)

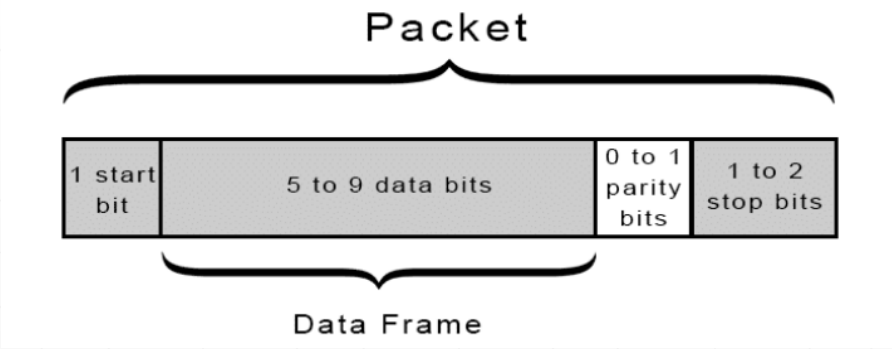


UART Data Frame

Instead of a clock signal, the transmitting UART adds start and stop bits to the data packet being transferred.



- Data frame
- One start bit
- Data (LSB first or MSB, and size of 7, 8, 9 bits)
- Optional parity bit
- One or two stop bit



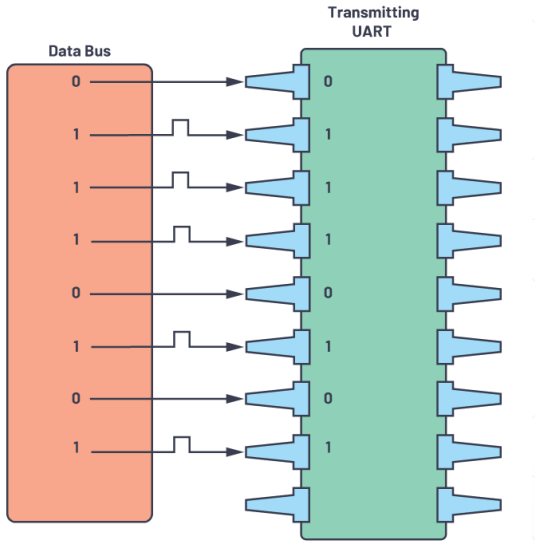
Baud Rate

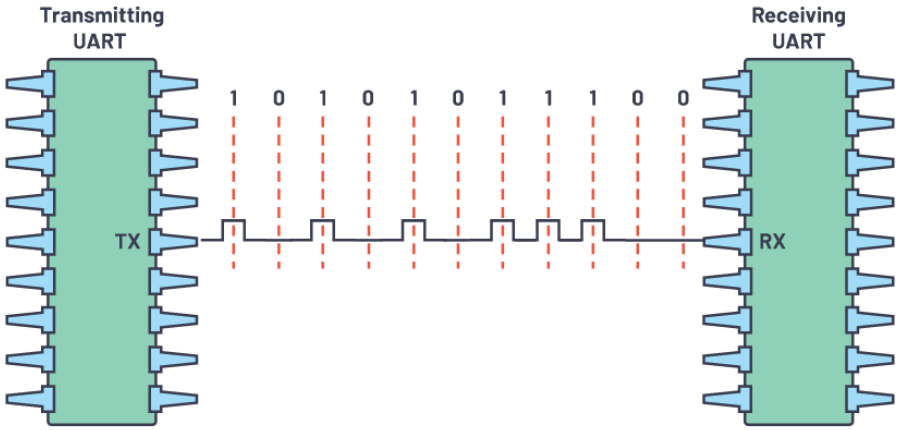
- Historically used in telecommunication to represent the number of pulses physically transferred per second
- In digital communication, baud rate is the number of bits physically transferred per second.
  - Example: Baud rate is 9600
  - Each frame: a start bit, 8 data bits, a stop bit, and no parity bit.
  - Transmission rate of actual data
    - ~~9600/8 = 1200 bytes/second~~
    - 9600/(1 + 8 + 1) = 960 bytes/second

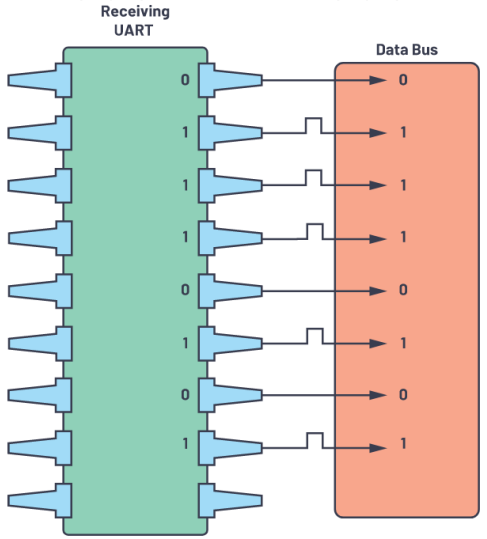
  The start and stop bits are the protocol overhead
- The maximum baud rate is 11520, usually used 9600 baud.

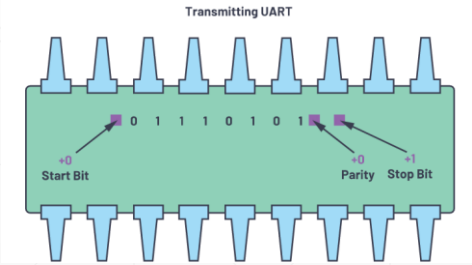
Steps of UART Transmission

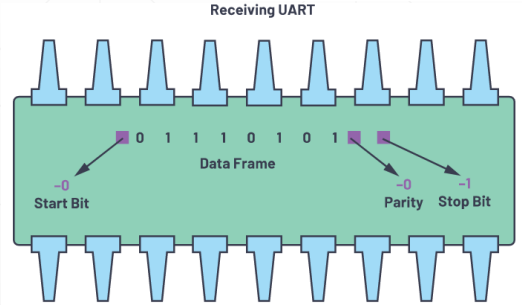
The transmitting UART receives/send data in parallel from the data bus:











Adds/drop the start bit, parity bit, and the stop bit(s) to the data frame

Advantages and Disadvantages of UARTs

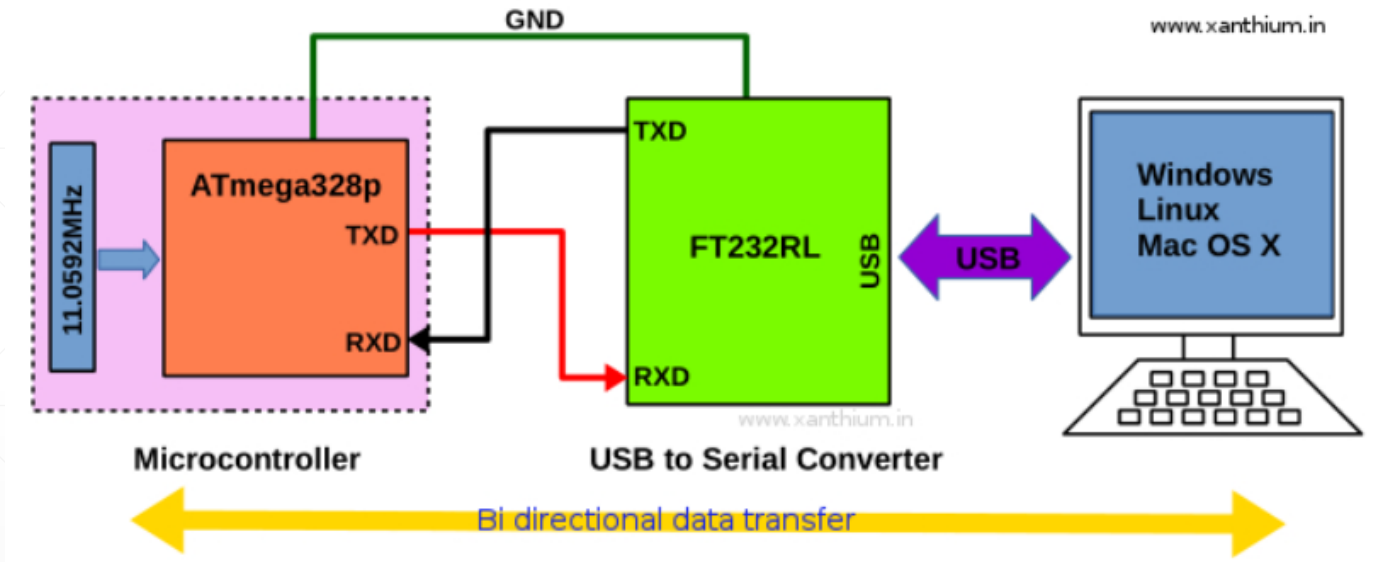
Advantages

- Only uses two wires
- No clock signal is necessary
- Has a parity bit to allow for error checking
- The structure of the data packet can be changed as long as both sides are set up for it
- Well documented and widely used method

Disadvantages

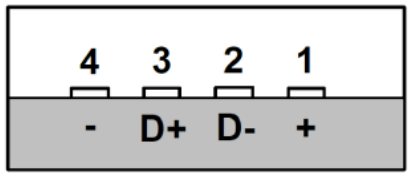
- The size of the data frame is limited to a maximum of 9 bits
- Doesn’t support multiple slave or multiple master systems
- The baud rates of each UART must be within 10% of each other

ATmega328P to PC Serial Communication using UART

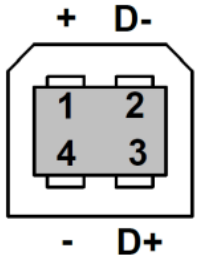


Source: https://www.xanthium.in/how-to-avr-atmega328p-microcontroller-usart-uart-embedded-programming-avrgcc

USB Layers



Standard A



Standard B

- Four shielded wires: two for power (+5V, ground), two for data (D+, D-)
- D+ and D- are twisted to cancel external electromagnetic interference

Tasks and Assignment

- Task 1: Controlling LED by Python code in the Computer through Arduino
- Task 2: Reading the Resistor’s Value from the Computer
- Task 3: Controlling LED based on the Resistor’s Value
- Lab Assignment: Controlling 5 LEDs based on the Resistor’s Value.

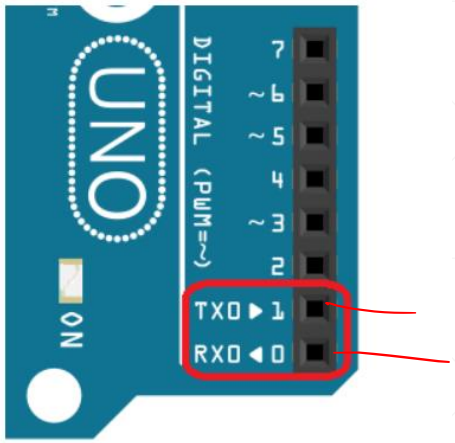
Arduino Serial Port

KMITL
พระจอมเกล้าลาดกระบัง

- Serial is used for communication between the Arduino board and a computer or other devices.
- All Arduino boards have at least one serial port (also known as a UART or USART): Serial.
- It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB.
- if you use these functions, you cannot also use pins 0 and 1 for digital input or output.

Arduino Serial Port

BOARD	USB CDC NAME	SERIAL PINS	SERIAL1 PINS	SERIAL2 PINS	SERIAL3 PINS
Uno, Nano, Mini		0(RX), 1(TX)			
Mega		0(RX), 1(TX)	19(RX), 18(TX)	17(RX), 16(TX)	15(RX), 14(TX)
Leonardo, Micro, Yún	Serial		0(RX), 1(TX)		
Uno WiFi Rev.2		Connected to USB	0(RX), 1(TX)	Connected to NINA	
MKR boards	Serial		13(RX), 14(TX)		
Zero	SerialUSB (Native USB Port only)	Connected to Programming Port	0(RX), 1(TX)		
Due	SerialUSB (Native USB Port only)	0(RX), 1(TX)	19(RX), 18(TX)	17(RX), 16(TX)	15(RX), 14(TX)
101	Serial		0(RX), 1(TX)		



Arduino uno board has **one serial port at digital pins 0(RX) and 1(TX)** to communicate with other external serial devices or with computer through USB cable.

Serial Function

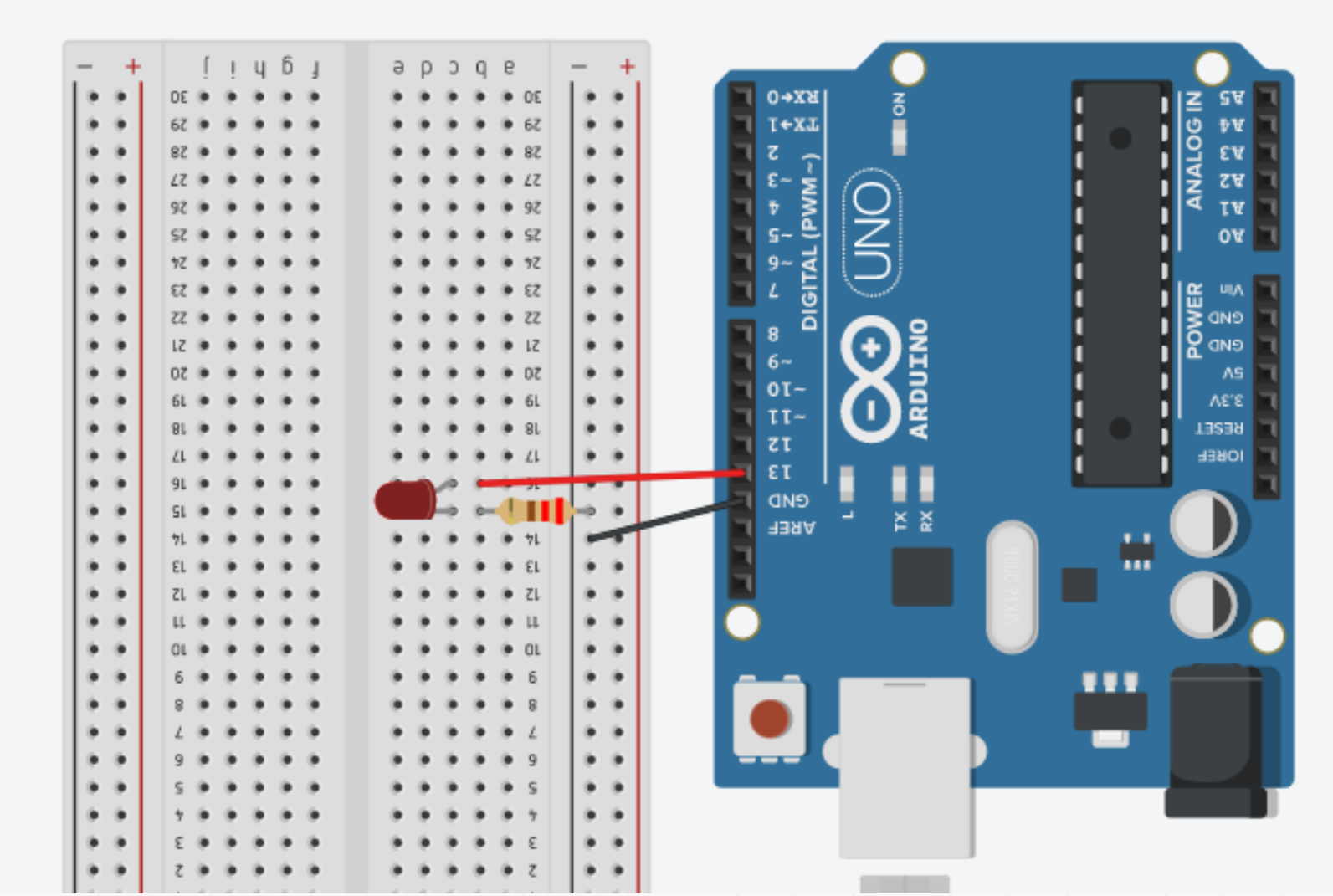
- if(Serial)
- available()
- availableForWrite()
- begin()
- end()
- find()
- findUntil()
- flush()
- parseFloat()
- parseInt()
- peek()
- print()
- println()
- read()
- readBytes()
- readBytesUntil()
- readString()
- readStringUntil()
- setTimeout()
- write()
- serialEvent()

Required Components

Components	Quantity
Arduino Uno and USB cable	1
LED 5mm (Red or Yellow)	2
Resistors 220R	2
Variable Resistor (Potentiometer)	1
Breadboard	1
Notebook Computer installed Arduino IDE and Python	1

Consult with Dr.Sumek Wisayataksin's Lab 3 : Analog Signal and Serial Interface from Intelligent Systems and Digital Device course

Task 1: Arduino Uno, LED and Resistor Circuit Connection



Arduino code for Controlling LED

Open Arduino IDE and write the code below.

```
// Send the control signal from Python to LED based on Analog Signal
const int LEDPin = 13;
char con_val = '0';

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

void loop() {
  // put your main code here, to ru

  if (Serial.available()>0)
  {
     con_val = char(Serial.read());
     LEDcontrol(con_val);
  }
  delay(100);
}
```

```
void LEDcontrol(char con)
{
  if (con == '1')
  {
      //Serial.write("From Arudion, LED is ON \n");  // send t
      Serial.print("From Arudion, LED is ON \n");  // send to
      digitalWrite(LEDPin, HIGH);   // turn the LED on (HIGH :

  }
  if (con == '0')
  {
      //Serial.write("From Arudion, LED is OFF \n");  // send
      Serial.print("From Arudion, LED is OFF \n");  // send to
      digitalWrite(LEDPin, LOW);    // turn the LED off by mal
  }
}
```

Control the LED from Arduino IDE's Serial Monitor

KMITL
พระจอมเกล้าลาดกระบัง

- Open Serial Monitor from Arduino IDE and observe the following questions

1. If the difference baud rates are set, is there any changes in Serial Monitor display? No
2. How many functions are there to read incoming serial data? One
3. How many functions are there to write outgoing serial data?

https://www.arduino.cc/reference/en/language/functions/communication/serial/

pySerial: Python Serial Port Library

KMITL
พระจอมเกล้าลาดกระบัง

- This module encapsulates the access for the serial port. It provides backends for Python running on Windows, OSX, Linux, BSD (possibly any POSIX compliant system) and IronPython. The module named “serial” automatically selects the appropriate backend.
- **Installation**: https://pyserial.readthedocs.io/en/latest/pyserial.html#installation
- **API**: https://pyserial.readthedocs.io/en/latest/pyserial_api.html
- Check the port by “python -m serial.tools.list_ports”
- **ShortIntro**: https://pyserial.readthedocs.io/en/latest/shortintro.html

Source: https://pyserial.readthedocs.io/en/latest/

Install PySerial and Control Arduino

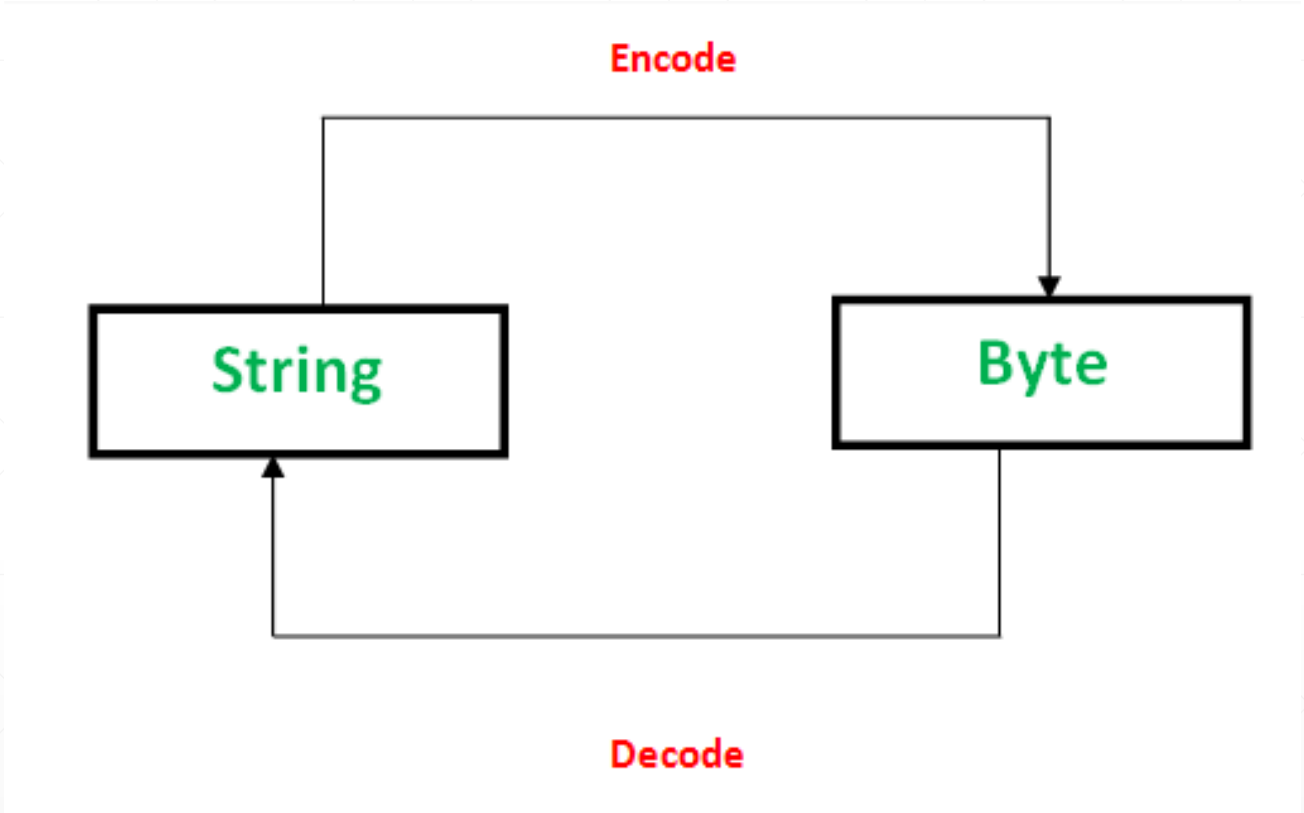
Open Command Prompt from your computer OS, and then install PySerial and type Python.

```
>>> import serial
>>> dev = serial.Serial("COMx“ or “/dev/ttyACMx“, baudrate=9600)
>>> dev.write(b'1')
>>> dev.write(b'0’)
>>> dev.write('0')
>>> dev.write(b'1’)
>>> print(dev.readline()) // repeat many time
>>> dev.close()
>>> dev = serial.Serial("COMx“ or “/dev/ttyACMx“, baudrate=9600, timeout = 0)
```

Repeat again.

Source: https://www.pythonforthelab.com/blog/how-control-arduino-computer-using-python/

Python Code for Receiving/Sending to Arduino Board



Check out about String and Byte and how they are different?

Control the LED from Computer using Python

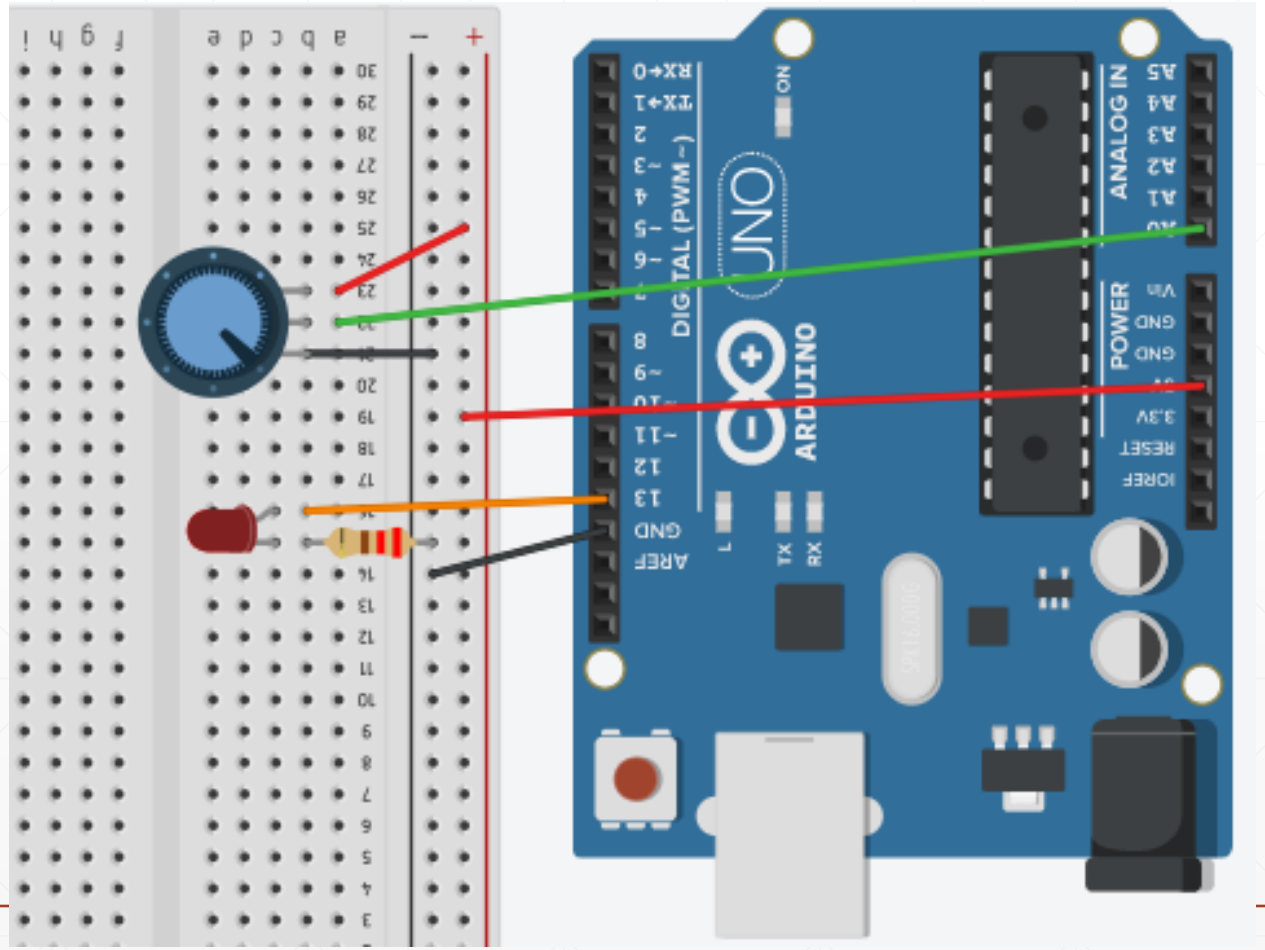
KMITL
พระจอมเกล้าลาดกระบัง

- Open Python IDE and write the below code

```
import serial
import sys
LED_flag = False
with serial.Serial('COM3',9600, timeout=1) as serArd:
    print(f"The Arduino board is connect through {serArd.port}")
    while True:
        try:
            con_val = input(f"Enter 1 for turn on LED and 0 for turn off LED : ")
            while not con_val in ['0', '1', 'q']:
                print(f"Please enter 1 or 0 !")
                con_val = input(f"Enter 1 for turn on LED and 0 for turn off LED : ")
            print(f'You entered {con_val}')
            if (serArd.writable() and con_val != 'q'):
                serArd.write(con_val.encode())
                myData = serArd.readline().decode()
                print(myData)

            if con_val == 'q':
                print('Program is stopped!')
                break

        except serial.SerialException as er:
            print(er)

        except KeyboardInterrupt:
            sys.exit(0)
```

Task 2: Reading the Resistor's Value from Computer

- Add a variable resistor (Potentiometer) into your circuit as below.



Reading the Value from Variable Resistor in Arduino Code

1. Add the Snippet below to LED control function in your Arduino Program

```
if (con == '2')
{
    double val =(analogRead(0)-512)/512.0;
    Serial.println(val);
}
```

2. Test with Arduino's Serial Monitor for reading the analog value as well as controlling LED.

3. Try to use the different functions: serial.write(), serial.print() and serial.println() in your code and check your result.

- https://www.arduino.cc/reference/en/language/functions/communication/serial/

Python Code for Receiving/Sending to Arduino Board

KMITL
พระจอมเกล้าลาดกระบัง

- Open the previous python program on Python IDE and modify your code to accept ‘2’.
- Run the program and test the result.

Reference

- https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/
- https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html
- https://www.xanthium.in/how-to-avr-atmega328p-microcontroller-usart-uart-embedded-programming-avrgcc
- Lecture Note Slides from ICEN 553/453– Fall 2018: Cyber-Physical Systems () by Prof. Dola Saha