

Practical Report

For IoT Practical



2022

Darshan Ramjiyani (DSP)

DOCS, KSKV Kachchh University

* **4.8 Serial Communication – Send Binary data from python and receive them in Arduino**

In this practical, the position of mouse pointer in (x, y) coordination will sent to the Arduino board. Arduino process this data and print the decimal value of sent data.

* **Python Code:**

from time import sleep

import pyautogui

import serial

PORT: str = "com9"

BAUD\_RATE: int = 9600

CONN: serial.Serial = None

def main():

    setup()

    while True:

        loop()

def setup():

    global BAUD\_RATE, CONN, PORT

    print("[Python] : Connecting Dwaidh terminal to Arduino. Please wait...")

    while True:

        try:

            CONN = serial.Serial(PORT, BAUD\_RATE)

            print(

                f"[Python] : Dwaidh terminal connected to the Arduino via {PORT} at {BAUD\_RATE} bps.")

            break

        except serial.SerialException as e:

            print(

                "[Python] : Can not connect to the port. Try again in 2 mininutes. ", e.args)

            from time import sleep

            sleep(2)

    while True:

        receive\_data: str = CONN.readline().decode("ascii")

        if len(receive\_data) > 0:

            if "Connected" in receive\_data:

                print("[Python] : Arduino has successfully verify the connection.")

                break

            else:

                print(f"[Arduino]: {receive\_data}")

        continue

def loop():

    while True:

        receive\_data: str = CONN.readline().decode("ascii")

        if len(receive\_data) > 0:

            if "Provide data" in receive\_data:

                x\_coordination, y\_coordination = pyautogui.position()

                print(f"[Python] : Sending ({x\_coordination},{y\_coordination}) as ({format(x\_coordination, '08b')}, "

                      f"{format(y\_coordination, '08b')}) to the Arduino.")

                ""

                CONN.write(bytes(""+format(x\_coordination, '08b')+","+format(y\_coordination, '08b'), "ascii"))

            elif "done" in receive\_data:

                print("[Arduino] : Process done.")

                x = input("[Python] : Press Y then enter to continue.")

                if x.lower() == 'y':

                    continue

                else:

                    print("[Python] : Closing the connection with Arduino...")

                    CONN.close()

                    print("[Python] : Connection has been closed.")

                    exit(0)

            else:

                print(f"[Arduino]: {receive\_data}")

        sleep(1)

if \_\_name\_\_ == '\_\_main\_\_':

    main()

* **Arduino Code:**

String data, subData1, subData2;

long binaryNumber;

void setup()

{

  int baudRate = 9600;

  /\* Established Serial Communication. \*/

  Serial.begin(baudRate);

  /\* Wait until Serial Communication not established. \*/

  while (!Serial)

  {

  }

  Serial.println("Connected.");

}

void loop()

{

  Serial.println("Provide data.");

  while (Serial.available() == 0)

  {

    ;

  }

  Serial.print("Data Recieved : ");

  data = Serial.readString();

  Serial.println(data);

  splitTheData();

  Serial.print("X-Coordination : ");

  Serial.print(convertBinaryToDecimal((long)subData1.toInt()));

  Serial.print(" Y-Coordination : ");

  Serial.println(convertBinaryToDecimal((long)subData2.toInt()));

  delay(1000);

  Serial.println("done");

}

void splitTheData()

{

  int sep\_pos = data.indexOf(",");

  subData1 = data.substring(0, sep\_pos);

  subData2 = data.substring(sep\_pos+1, data.length());

}

long convertBinaryToDecimal(long binary) {

  long number = binary;

  long decimalVal = 0;

  long baseVal = 1;

  long tempVal = number;

  long previousDigit;

  while (tempVal) {

    //Converts Binary to Decimal

    previousDigit = tempVal % 10;

    tempVal = tempVal / 10;

    decimalVal += previousDigit \* baseVal;

    baseVal = baseVal \* 2;

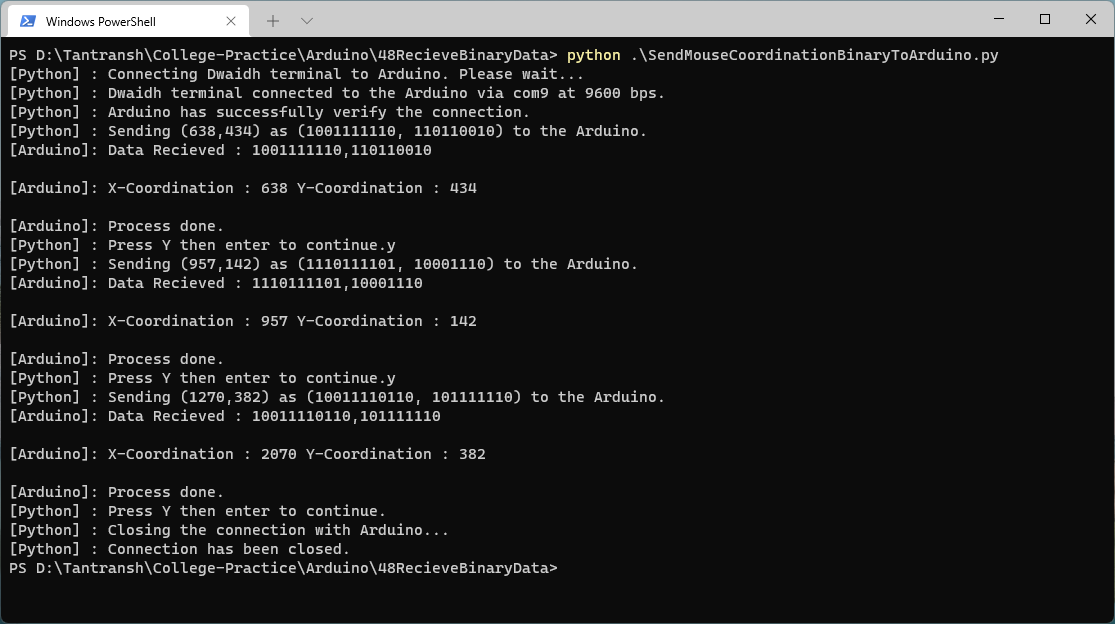
  }

  //Returns the Decimal number

  return decimalVal;

}

* **Output:**

****