Assignment #2 Documentation

Robert Schmidt

Dr. Vincent

9/22/2021

1. Can you adjust data transfer rates on I2C? What transfer rates are available?
   Yes, I2C transfer rates are: 3.4Mbps(high speed), 1Mbps(fast mode plus), 400kbps(fast mode), 100kbps(standard).
2. What is the maximum rate of data transfer that you can achieve with an I2C setup?
   Maximum rate of data transfer is 3.4Mbps(high speed)
3. What Arduino library do you need to use to implement this?
   The wire library >>>> #include Wire.h
4. How many bytes can you transmit in one transaction using the Arduino?
   32 bytes, after reading the Arduino can be altered to handle more by changing some settings but I believe that 32 bytes would be the standard answer to this question

Arduino Code for Assignment 2 Exercise 1

```
// This code will establish 2-way communication between the Arduino and Pi. The Pi will request an integer value from the user and send it to the Arduino. The Arduino will add 5 to the number and send it back to the Pi.

//You will need an Arduino, pi, computer screen, and the required wires in order to connect your board the correct way

#include <Wire.h>

#define SLAVE_ADDRESS 0x04

int number = 0;

int state = 0;


void setup() {

  pinMode(13, OUTPUT);

  Serial.begin(115200); // start serial for output

  // initialize i2c as slave

  Wire.begin(SLAVE_ADDRESS);


  // define callbacks for i2c communication
```

```
  Wire.onReceive(receiveData);
  Wire.onRequest(sendData);
//print ready to monitor
  Serial.println("Ready!");
}

void loop() {
  delay(100);
}

// callback for received data
void receiveData(int byteCount)
{ Serial.print("data received: ");
//while loop to read incoming data
  while (Wire.available()) {
//setting number equal to the incoming data
    number = Wire.read();
//Adding 5 to number that was received
    number += 5;
//print number to monitor
    Serial.print(number);
    Serial.print(" ");
  }
  Serial.println(" ");
}

// callback for sending data
void sendData() {
  //sending number
```

```
  Wire.write(number);
}
```

Arduino Code for Assignment 2 Exercise 2

//This code will create a communication link between the Pi and Arduino. The Pi will ask for both a value and an offset, and then executes a bus.write_byte_data. The Arduino saves the value to a different variable depending on the offset. The Pi then asks for an offset to be read, and executes abus.read_byte_data. If I2c reads a 0 for the offset 5 will be added to the number, if I2C reads a 1 for the offset then 10 will be added to the number.

//You will need an Arduino, pi, computer screen, and the required wires in order to connect your board the correct way

```
//import libraries
#include <Wire.h>

#define SLAVE_ADDRESS 0x04
//initialize variables
int number = 0;
int state = 0;
byte data[32] = 0;

void setup() {
  pinMode(13, OUTPUT);
  Serial.begin(115200); // start serial for output
  // initialize i2c as slave
  Wire.begin(SLAVE_ADDRESS);

  // define callbacks for i2c communication
  Wire.onReceive(receiveData);
  Wire.onRequest(sendData);
```

```
  Serial.println("Ready!");
}

void loop() {
  delay(100);
}

// callback for received data
void receiveData(int byteCount)
{ Serial.print("data received: ");
  int i = 0;
  while (Wire.available()) {
    data[i] = Wire.read();
//to monitor
    Serial.print(data[i]);
    Serial.print(" ");
    i++;
  }
  i--;
//if 0 add 5
  if (data[0] == 0){
    data[i] += 5;
  }
//if 1 add 10
  else if (data[0] == 1){
    data[i] += 10;
  }
//set number equal to date for wire later
  number = data[i];
```

```
  Serial.println(" ");
}


// callback for sending data
void sendData() {
  //write data
  Wire.write(number);
}
```

Arduino Code for Assignment 2 Exercise 3

//This code is an extension of Exercise 2, the difference here is the implementation of the LCD screen which will print out the number it Got: and the number Sent:

//You will need an Arduino, pi, computer screen, LCD screen, and the required wires in order to connect your board the correct way

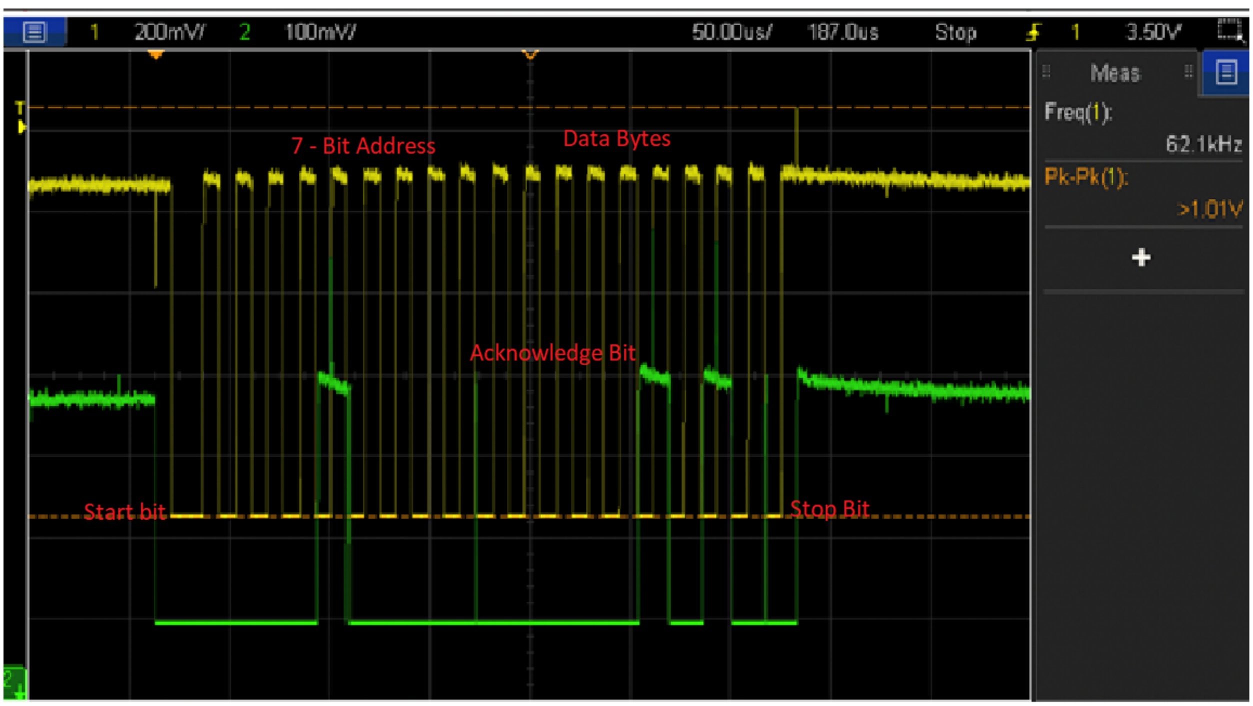
```
#include <Wire.h>


#define SLAVE_ADDRESS 0x04
//initialize variables
int number = 0;
int state = 0;
byte data[32] = 0;


void setup() {
  pinMode(13, OUTPUT);
  Serial.begin(115200); // start serial for output
  // initialize i2c as slave
  Wire.begin(SLAVE_ADDRESS);
```

```
  // define callbacks for i2c communication
  Wire.onReceive(receiveData);
  Wire.onRequest(sendData);
//to monitor
  Serial.println("Ready!");
}

void loop() {
  delay(100);
}

// callback for received data
void receiveData(int byteCount)
{ Serial.print("data received: ");
  int i = 0;
  while (Wire.available()) {
//reading in data
    data[i] = Wire.read();
//to monitor
    Serial.print(data[i]);
    Serial.print(" ");
    i++;
  }
  i--;
//if 0 add 5
  if (data[0] == 0){
    data[i] += 5;
  }
//if 1 add 10
```

```
 else if (data[0] == 1){
   data[i] += 10;
 }
//set number to data wire later
 number = data[i];
 Serial.println(" ");
}


// callback for sending data
void sendData() {
 //writing data
 Wire.write(number);
}
```

Oscilloscope graph Assignment 2 Exercise 4:

// Here the Oscilloscope was used to scope out the SCL and SDA lines on the same graph. This is done in order to see the transmission of one 8 bit data from the leader to the follower. As shown the start bit, the 7 bit address, acknowledge bit, data byte, and stop bit have been marked.



Arduino Code for Assignment 2 Exercise 5

//write_I2C_block_data and read_I2C_data will be implemented in this code to send multiple bytes back and fourth between the pi and Arduino. The pi will ask the user for a string and then convert it to integer values and send those values to the Arduino. The Arduino well then reverse this string and send it back to the Pi where the pi will convert the integer value back into a string and then print the string.

//You will need an Arduino, pi, computer screen, and the required wires in order to connect your board the correct way

```
#include <Wire.h>


#define SLAVE_ADDRESS 0x04
//initializing variables
int number = 0;
int state = 0;
byte data[32] = {0};


void setup() {
  pinMode(13, OUTPUT);
//setting baud rate and begin program
  Serial.begin(115200); // start serial for output
  // initialize i2c as slave
  Wire.begin(SLAVE_ADDRESS);


  // define callbacks for i2c communication
  Wire.onReceive(receiveData);
  Wire.onRequest(sendData);


  Serial.println("Ready!");
}
```

```
void loop() {
  delay(100);
}

// callback for received data
void receiveData(int byteCount)
{
//to monitor
  Serial.print("data received: ");
  int i = 0;
  while (Wire.available()) {
//bringing in the data
    data[i] = Wire.read();
//to monitor
    Serial.print(data[i]);
    Serial.print(" ");
    i++;
  }
//the following loop reverses the data
  int k = i;
  int temp;
  for(i = 0; i<k; i++, k--){
    temp = data[i];
    data[i] = data[k];
    data[k] = temp;
  }
  Serial.println(" ");
}
```

```
// callback for sending data
void sendData() {
//writing data with length 32
  Wire.write(data,32);
}
```

Arduino Code for Assignment 2 Exercise 6

//In this code a potentiometer was hooked up to the Arduino where the Arduino read the voltage across the potentiometer. The Arduino will then send this information to the pi where the pi will display the voltages to a LCD screen.

//You will need an Arduino, pi, computer screen, LCD screen, and the required wires in order to connect your board the correct way

```
//Basic USB serial implementation on an Arduino
//initialize variables
int sensorPin = A0;
int sensorValue = 0;


void setup() {
  Serial.begin(115200); // start serial for output
}


void loop() {
//ensure the value is between 0-255
  sensorValue = (analogRead(sensorPin)- 3)/4;
    Serial.println(sensorValue);
}
void serialEvent() {
  if (Serial.available() > 0) {
//reading data
    sensorValue = Serial.read();
```

```
    sensorValue = true;
  }
//ensure transmitter finished sending
  Serial.flush();
}
```

My raspberry pi code for this section has disappeared but my Arduino code for the exercise you will see further up in this document. Both the Exercise 6 codes (Arduino and pi) were shown to Prof. McSweeney during my Demo. During my demo we had to unplug my pi/Arduino due to it freezing and were still able to find my raspberry pi code saved for assignment 2 exercise 6 so I am unsure why it was deleted and or disappeared. My code was able to read the potentiometer on the Arduino and was limited between 0-255 like the exercise had asked for. My code lacked the ability to send this information to the raspberry pi in order for the voltages to be displayed on the LCD. Admittedly, I ran out of time and exercise 6 was the last code I left and was unable to complete it. Prof. McSweeney ended up confirming that exercise 6 was the one code I lacked on in this assignment and my demonstration grade reflects that. My hopes are that leniency will be shown when grading this exercise 6 due to this weird error that has arisen and that Prof. McSweeney could confirm that the code indeed did exist.

Arduino Code for Assignment 2 Exercise 7-8

//included in this code is exercise 7 and exercise 8. This code is capable of handling communication errors known as an "IOError". use exception handling was implemented in order to prevent the system from stopping when encountering a error such as removing the SDA pin. If this does occur the error will be printed to the LCD screen for the user to see. Exercise 1 was repeated using Serial communication which reflects in the below code.

//You will need an Arduino, pi, computer screen, LCD screen, and the required wires in order to connect your board the correct way

```
//Basic USB serial implementation on an Arduino
int data;
String temp;
bool DataRead;
void setup() {
//setting baud rate and begin program
  Serial.begin(115200); // start serial for output
```

```
}
void loop() {
  if (DataRead) {
    //adding 5 to value inside of data
    data += 5;
//printing data monitor
    Serial.println(data);
    DataRead = false;
  }
}
void serialEvent() {
  if (Serial.available() > 0) {
    data = Serial.read();
    DataRead = true;
  }
  Serial.flush();
}
```

Rasberry Pi code Assignment 2 Exercise 1:

// This code will establish 2-way communication between the Arduino and Pi. The Pi will request an integer value from the user and send it to the Arduino. The Arduino will add 5 to the number and send it back to the Pi.

//You will need an Arduino, pi, computer screen, and the required wires in order to connect your board the correct way

```
//importing libraries
import smbus
import time
# for RPI version 1, use "bus = smbus.SMBus(0)"
bus = smbus.SMBus(1)
```

```
# This is the address we setup in the Arduino Program
address = 0x04
//defining writeNumber function
def writeNumber(value):
#bus.write_byte(address, value)
    bus.write_byte_data(address, 0, value)
    return -1
//defining readNumber function
def readNumber():
    number = bus.read_byte(address)
    #number = bus.read_byte_data(address, 0)
    return number
//asking user for number 1-9 in a while loop
while True:
    var = input("Enter 1 – 9: ")
    var = int(var)
//breaking out of the while loop
    if not var:
        continue

    writeNumber(var)
    print ("RPI: Hi Arduino, I sent you ", var)
    # sleep one second
    time.sleep(1)

    number = readNumber()
    print ("Arduino: Hey RPI, I received a digit ", number)
```

Rasberry Pi code Assignment 2 Exercise 2:

//This code will create a communication link between the Pi and Arduino. The Pi will ask for both a value and an offset, and then executes a bus.write_byte_data. The Arduino saves the value to a different variable depending on the offset. The Pi then asks for an offset to be read, and executes abus.read_byte_data. If I2c reads a 0 for the offset 5 will be added to the number, if I2C reads a 1 for the offset then 10 will be added to the number.

//You will need an Arduino, pi, computer screen, and the required wires in order to connect your board the correct way

```
//importing libraries

import smbus

import time

# for RPI version 1, use “bus = smbus.SMBus(0)”

bus = smbus.SMBus(1)


# This is the address we setup in the Arduino Program

address = 0x04

//defining the writeNumber function

def writeNumber(value):

#bus.write_byte(address, value)

   bus.write_byte_data(address, 0, value)

   return -1

//defining the readNumber function

def readNumber():

   number = bus.read_byte(address)

   #number = bus.read_byte_data(address, 0)

   return number

//asking user for number 1-9

while True:

   var = input("Enter 1 – 9: ")

   var = int(var)
```

```
//asking user for an offset
   numOffset = input ("Please enter a offset value of 0 or 1: ")
   numOffset = int(numOffset)
//breaking out of the while loop
   if not var:
      continue
   //if offset received is 0 go here
   if numOffset == 0:
      bus.write_byte_data(address, 0, var)
      time.sleep(1)
      number = bus.read_byte(address)
  //if offset received is 1 go here
   if numOffset == 1:
      bus.write_byte_data(address, 1, var)
      time.sleep(1)
      number = bus.read_byte(address)

   writeNumber(var)
   print ("RPI: Hi Arduino, I sent you ", var)
   # sleep one second
   time.sleep(1)

   print ("Arduino: Hey RPI, based on your number and offset I am sending back a digit ", number)
```

<u>Rasberry Pi code Assignment 2 Exercise 3:</u>

//This code is an extension of Exercise 2, the difference here is the implementation of the LCD screen which will print out the number it Got: and the number Sent:

//You will need an Arduino, pi, computer screen, LCD screen, and the required wires in order to connect your board the correct way

```
//importing libraries
import time
import board
import adafruit_character_lcd.character_lcd_rgb_i2c as character_lcd
import smbus

#LCD display
# Modify this if you have a different sized Character LCD
lcd_columns = 16
lcd_rows = 2

# Initialise I2C bus.
i2c = board.I2C()  # uses board.SCL and board.SDA

# Initialise the LCD class
lcd = character_lcd.Character_LCD_RGB_I2C(i2c, lcd_columns, lcd_rows)

lcd.clear()

# for RPI version 1, use “bus = smbus.SMBus(0)”
bus = smbus.SMBus(1)

# This is the address we setup in the Arduino Program
address = 0x04

def writeNumber(value):
#bus.write_byte(address, value)
   bus.write_byte_data(address, 0, value)
```

```
    return -1

def readNumber():
    number = bus.read_byte(address)
    #number = bus.read_byte_data(address, 0)
    return number

//asking user for a number 1-9
while True:
    var = input("Enter 1 – 9: ")
    var = int(var)
// asking user for a offset value
    numOffset = input ("Please enter a offset value of 0 or 1: ")
    numOffset = int(numOffset)
    if not var:
        continue
    //will go here if numOffset is zero
    if numOffset == 0:
        bus.write_byte_data(address, 0, var)
        time.sleep(1)
        number = bus.read_byte(address)
    //will go here if numOffset is one
    if numOffset == 1:
        bus.write_byte_data(address, 1, var)
        time.sleep(1)
        number = bus.read_byte(address)

    writeNumber(var)
    print ("RPI: Hi Arduino, I sent you ", var)
```

```
    # sleep one second
    time.sleep(1)

    print ("Arduino: Hey RPI, based on your number and offset I am sending back a digit ", number)



    # Set LCD color to red
    lcd.color = [100, 0, 0]
    time.sleep(1)
//printing two line message to LCD with Var, Number
    while True:
        lcd.message = "Sent: %d \nGot: %d"%(var, number)
        break
    # Printing two line message
    #lcd.message = "sent: ", var
    #lcd.message = "got: ", number
    #lcd.message = "Sent: ", var
    #wait time 2s
    time.sleep(3)
    # Set LCD color to blue
    lcd.color = [0, 100, 0]
    time.sleep(2)
    # Set LCD color to green
    lcd.color = [0, 0, 100]
    time.sleep(2)
    # Set LCD color to purple
    lcd.color = [50, 0, 50]
    time.sleep(2)
    lcd.clear()
```

lcd.message = "Going to sleep\nCya later!"

time.sleep(5)

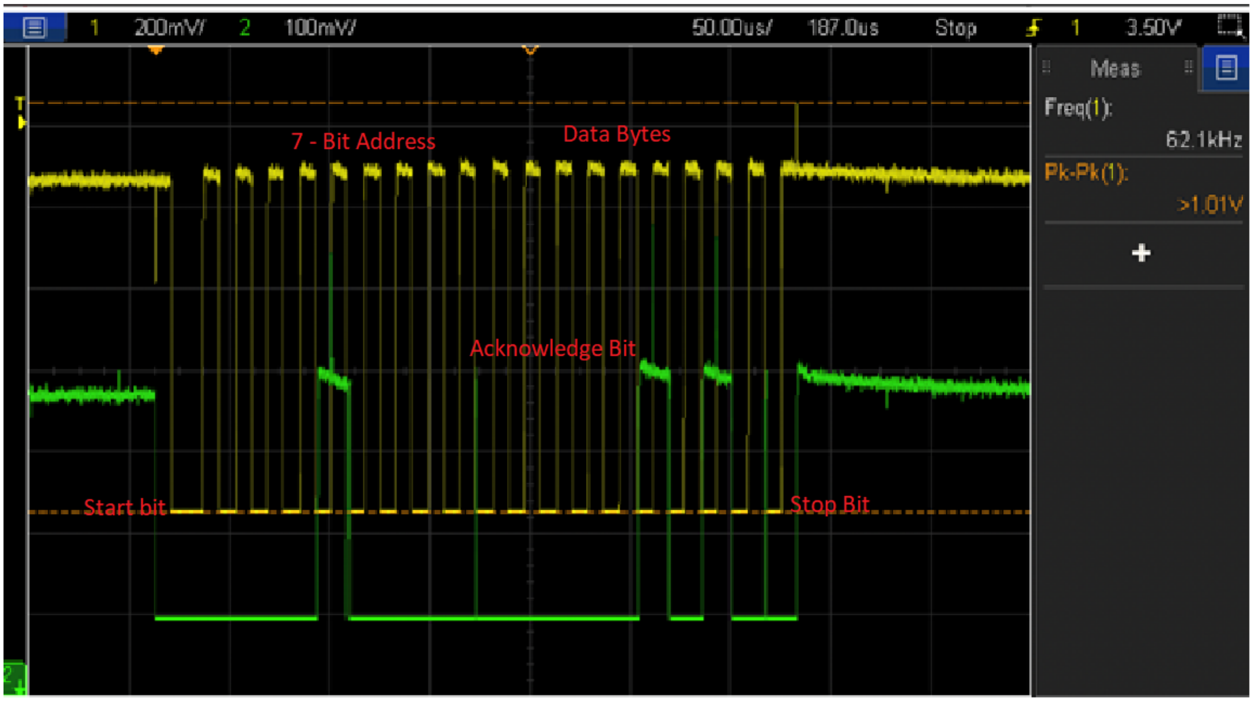
Turn off LCD backlights and clear text

lcd.color = [0, 0, 0]

lcd.clear()

Oscilloscope graph Assignment 2 Exercise 4:

// Here the Oscilloscope was used to scope out the SCL and SDA lines on the same graph. This is done in order to see the transmission of one 8 bit data from the leader to the follower. As shown the start bit, the 7 bit address, acknowledge bit, data byte, and stop bit have been marked.



Rasberry Pi code Assignment 2 Exercise 5:

//write_I2C_block_data and read_I2C_data will be implemented in this code to send multiple bytes back and fourth between the pi and Arduino. The pi will ask the user for a string and then convert it to integer values and send those values to the Arduino. The Arduino well then reverse this string and send it back to the Pi where the pi will convert the integer value back into a string and then print the string.

//You will need an Arduino, pi, computer screen, and the required wires in order to connect your board the correct way

```
//importing libraries
import time
import board
import smbus
import numpy
import adafruit_character_lcd.character_lcd_rgb_i2c as character_lcd

# Modify this if you have a different sized Character LCD
lcd_columns = 16
lcd_rows = 2

# Initializing the I2C bus.
i2c = board.I2C()  # uses board.SCL and board.SDA

# Initializing the LCD class
lcd = character_lcd.Character_LCD_RGB_I2C(i2c, lcd_columns, lcd_rows)

lcd.clear()

i = 0
# for RPI version 1, use “bus = smbus.SMBus(0)”
bus = smbus.SMBus(1)

# This is the address we setup in the Arduino Program
address = 0x04

def writeNumber(string):
#bus.write_byte(address, value)
   #bus.write_byte_data(address, 0, value)
```

```
    bus.write_i2c_block_data(address, 0, string)
    return -1

def readNumber():
    data = bus.read_i2c_block_data(address, 0)
    #number = bus.read_byte(address)
    #number = bus.read_byte_data(address, 0)
    return data

//getting input from user
while True:
    string = input("Enter a string: ")
    string = [ord(char) for char in string]

    if not string:
        continue
    //error handling
    try:
        writeNumber(string)

        print ("RPI: Hi Arduino, I sent you ", string)
        #data = [chr(char) for char in data]
        data = readNumber();
        print ("Arduino: Hey RPI, based on your input I have: ", data)
    except:
        print("I2C Error")
        # Set LCD color to red
        lcd.color = [100, 0, 0]
        time.sleep(1)
```

```
# Print two line message
lcd.message = "I2C Error"
lcd.clear()
```

Rasberry Pi code Assignment 2 Exercise 6:

//In this code a potentiometer was hooked up to the Arduino where the Arduino read the voltage across the potentiometer. The Arduino will then send this information to the pi where the pi will display the voltages to a LCD screen.

//You will need an Arduino, pi, computer screen, LCD screen, and the required wires in order to connect your board the correct way

My raspberry pi code for this section has disappeared but my Arduino code for the exercise you will see further up in this document. Both the Exercise 6 codes (Arduino and pi) were shown to Prof. McSweeney during my Demo. During my demo we had to unplug my pi/Arduino due to it freezing and were still able to find my raspberry pi code saved for assignment 2 exercise 6 so I am unsure why it was deleted and or disappeared. My code was able to read the potentiometer on the Arduino and was limited between 0-255 like the exercise had asked for. My code lacked the ability to send this information to the raspberry pi in order for the voltages to be displayed on the LCD. Admittedly, I ran out of time and exercise 6 was the last code I left and was unable to complete it. Prof. McSweeney ended up confirming that exercise 6 was the one code I lacked on in this assignment and my demonstration grade reflects that. My hopes are that leniency will be shown when grading this exercise 6 due to this weird error that has arisen and that Prof. McSweeney could confirm that the code indeed did exist.

Rasberry Pi code Assignment 2 Exercise 7-8:

//included in this code is exercise 7 and exercise 8. This code is capable of handling communication errors known as an "IOError". use exception handling was implemented in order to prevent the system from stopping when encountering a error such as removing the SDA pin. If this does occur the error will be printed to the LCD screen for the user to see. Exercise 1 was repeated using Serial communication which reflects in the below code.

//You will need an Arduino, pi, computer screen, LCD screen, and the required wires in order to connect your board the correct way

```
//import libraries
import serial
import time
#Setting the address
ser = serial.Serial('/dev/ttyACM0', 115200)
#Waiting for the connection to complete
time.sleep(3)
#Function that reads serial
def ReadfromArduino():
    while (ser.in_waiting > 0):
//error handling
        try:
            line = ser.readline().decode('utf-8').rstrip()
            print("serial output : ", line)
        except:
            print("Communication Error")
#sending a string
value = input("Enter a integer" + "\n")
#Remeber to encode the string to bytes
ser.write(value.encode())
# wait for Arduino to set up response
time.sleep(2)
ReadfromArduino()
print("Done")
```