Lab Worksheet 4

CSE461: Introduction to Robotics
Department of Computer Science and Engineering

Lab 04: Interfacing Arduino and Raspberry Pi using UART Protocol to Display IR Sensor Values on an I2C LCD

I. Topic Overview

This lab introduces the fundamentals of cross-platform microcontroller interfacing using the Arduino Uno and Raspberry Pi through the UART communication protocol. Students will learn how to establish serial communication between the two devices, enabling real-time data exchange. The lab will focus on reading sensor values from an IR sensor connected to the Arduino and transmitting the data to the Raspberry Pi. The Raspberry Pi will then display the received values on an I2C LCD display. By the end of this lab, students will develop a functional system integrating both microcontrollers for seamless data transmission and display.

II. Learning Outcome

After this lab, students will be able to:

1. Interface Arduino and Raspberry PI using UART protocol.
2. Design circuits that address voltage compatibility between 3.3V and 5V systems.
3. Develop code for real-time sensor data transmission from Arduino to Raspberry Pi.
4. Utilize the I2C protocol to display received sensor values on an LCD.
5. Critically troubleshoot hardware and software integration issues.

III. Materials

- Arduino Uno R3
- Raspberry PI 4
- Analog 6-array IR sensor
- I2C LCD Display (16x2)
- Resistor (220Ω)
- Breadboard
- Jumper wires

IV. UART Protocol Overview

UART (Universal Asynchronous Receiver-Transmitter) is a serial communication protocol that enables data exchange between microcontrollers without needing a clock signal. It uses two main lines for communication: TX (Transmit) and RX (Receive). Data is sent **byte-by-byte** in an asynchronous manner, making it a simple and efficient method for microcontroller communication. UART Pins on Arduino and Raspberry PI are as follows:

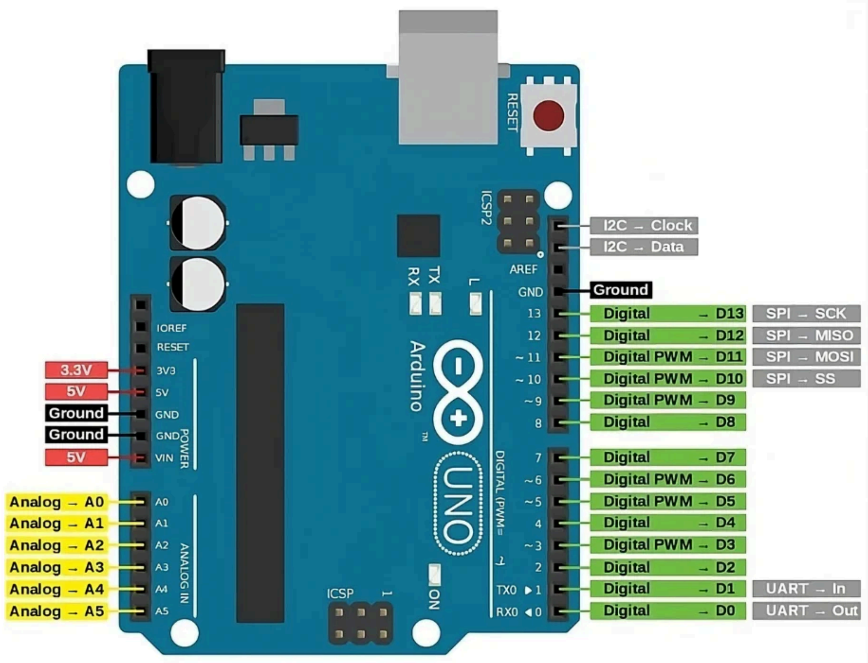
- **Arduino Uno:**
  - TX (Transmit) → Pin 1
  - RX (Receive) → Pin 0
- **Raspberry PI 4 (GPIO Header):**
  - TX (Transmit) → GPIO14 (Pin 8)
  - RX (Receive) → GPIO15 (Pin 10)

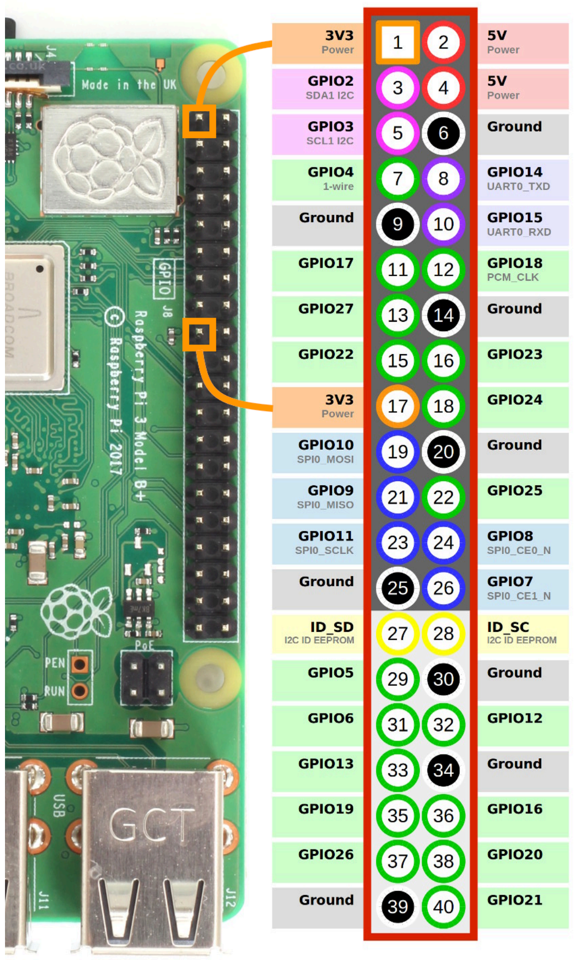
For successful communication, the **TX pin of one device must be connected to the RX pin of the other**, and vice versa.

V. I2C Protocol Overview

I2C (Inter-Integrated Circuit) is a synchronous, multi-device communication protocol that allows multiple master and slave devices to communicate using only two shared lines: SDA (Serial Data Line) which transfers data between devices, and SCL (Serial Clock Line) which synchronizes the data transfer. Unlike UART, which is a point-to-point communication protocol, I2C supports multiple devices on the same bus using unique 7-bit or 10-bit addresses assigned to each device. The master device (in this case, the Raspberry Pi) initiates communication and sends instructions to slave device (such as the LCD display). I2C pins on Arduino and Raspberry Pi are as follows:

- **Arduino Uno:**
  - SDA → A4
  - SCL → A5
- **Raspberry Pi 4 (GPIO Header):**
  - SDA → GPIO2 (Pin 3)
  - SCL → GPIO3 (Pin 5)





VI. Setting Up the Arduino IDE in Raspberry PI

If the Arduino IDE is not installed in your Raspberry Pi, please refer back to Lab Worksheet 3 :

CSE461 Lab Worksheet 3

VII. Enabling UART and I2C on Raspberry PI

- Open the terminal on the Raspberry PI.
- Run the following command to enable UART:

```
sudo raspi-config
```

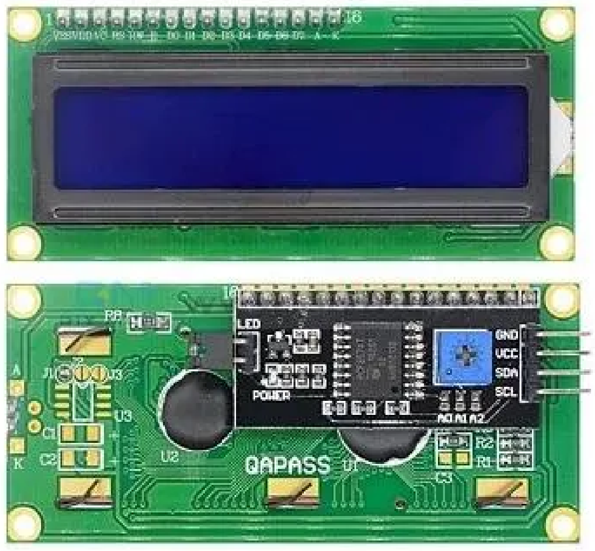
- Navigate to **Interfacing Options → Serial Port.**
- Disable the login shell over serial and enable the serial port hardware.
- Navigate to **Interfacing Options → I2C.**
- Enable the I2C communication protocol.
- Reboot the Raspberry Pi:

```
sudo reboot
```

VIII. Experiment 1 : Setting up an I2C LCD Display on Raspberry Pi

1. **Description of the components:**

   - Jumper Wires
   - 16*2 LCD (Liquid Crystal Display) with I2C adapter



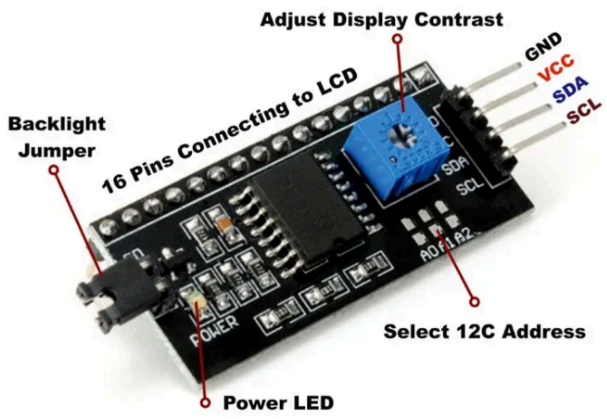
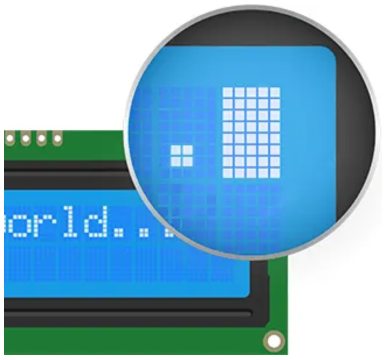


Fig. LCD with I2C module

A Liquid Crystal Display (LCD) is an electronic display module that uses liquid crystal technology to display characters, numbers, and symbols. The **16×2 LCD** name refers to its **character display format**:

- ➢ **16 columns** → It can display **16 characters per row**
- ➢ **2 rows** → It has **2 rows of characters**

Each character is displayed using a **5×8** pixel matrix, allowing clear representation of text and numbers.



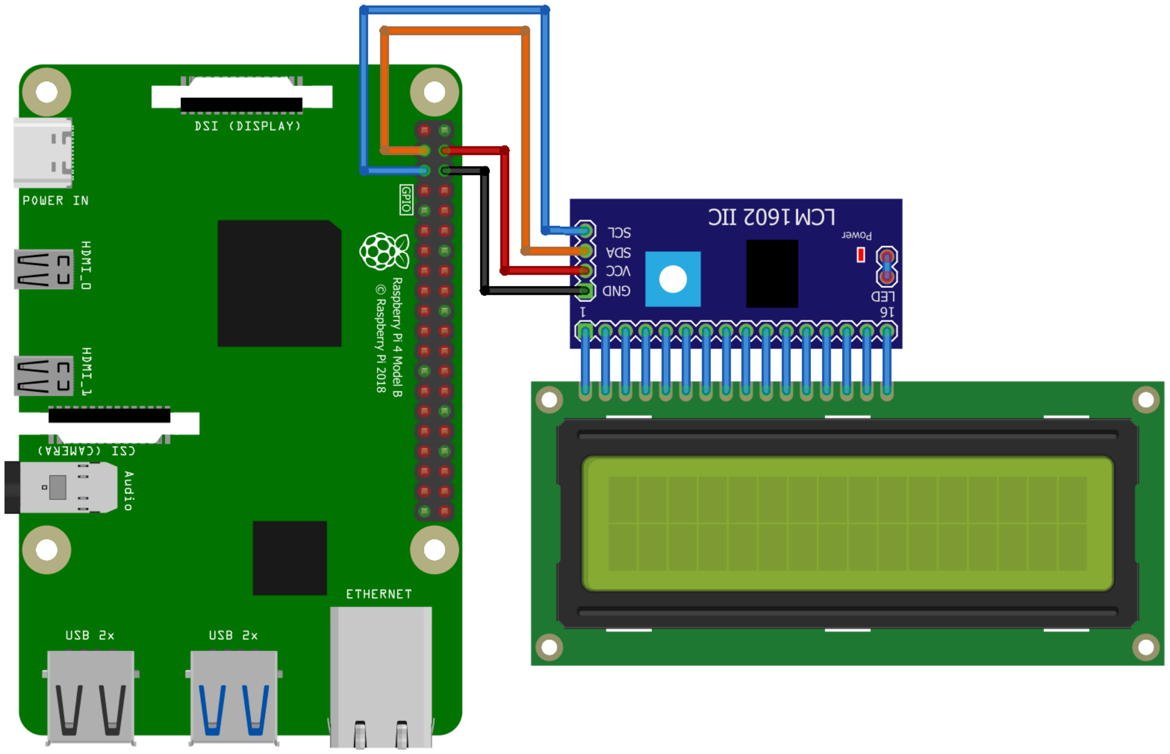
Why use an I2C module with LCD?

A standard 16×2 LCD requires at least 6 GPIO pins to function (RS, RW, E, D4–D7). To reduce wiring complexity, an I2C module (**such as PCF8574**) is used, which enables communication using only two pins (SDA and SCL)

2. **Setting up the circuit:**

- Connect VCC and GND of the I2C module to 5v and GND of RPi respectively.
- Connect SDA (LCD) to Pin 3 (GPIO2) and SCL (LCD) to Pin 5 (GPIO3).

3. **Circuit diagram:**



4. **Code:**

If you have enabled the I2C protocol successfully, write the following commands in the terminal before running the code:

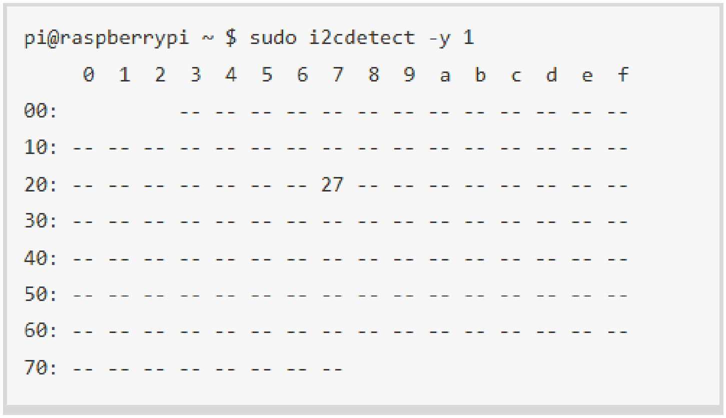
```
sudo apt install -y i2c-tools python3-smbus
```

```
pip install rpi-lcd --break-system-packages
```

After executing these commands, check if your Raspberry Pi properly detected the LCD display or not using the following command:

```
sudo i2cdetect -y 1
```

If detected properly, you should see the following in your terminal.



```
pi@raspberrypi ~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- 27 -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

The mentioned number is the I2C address of the LCD (usually 0x27 or 0x3F)

Note: If you do not see any address, try changing the wires first and then rest of the components

After setting everything up, now you can execute the following code in your Python IDE.

Raspberry PI Code:

```
from rpi_lcd import LCD
from time import sleep

lcd = LCD()

lcd.text('Hello World!', 1) #print the text in line 1
lcd.text('Raspberry Pi', 2) #print the text in line 2

sleep(10)
lcd.clear()
```

__How this works:__ At the hardware level, the Raspberry Pi communicates with the I2C LCD module using the SDA (GPIO2) and SCL (GPIO3) pins. The PCF8574 I2C module converts I2C signals into control signals for the LCD driver, which maps characters onto the 16×2 display. At the software level, lcd.text('Hello World!', 1) sends an I2C command to print text on the first row, while lcd.text('Raspberry Pi', 2) does the same for the second row. After a 10-second delay, lcd.clear() resets the display by sending a clear command to the LCD driver

.

IX. Experiment 2 : Setting up the 6-array IR sensor on Arduino Uno

1. **Description of the components:**

- Jumper Wires
- Analog 6-array IR sensor

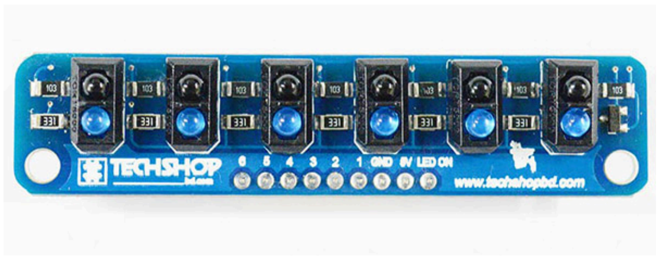
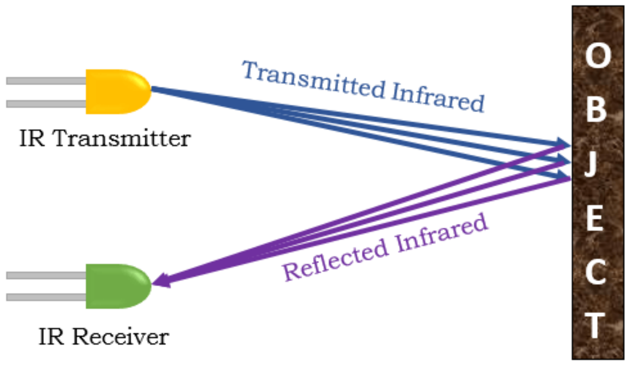


Fig. Analog 6-array IR sensor

This analog IR sensor is designed for Arduino Uno and other 5V Arduino boards, making it ideal for line follower and maze solver robots. It features 6 IR sensors with 1 cm spacing, covering 9 cm for precise line detection. The IR transmitter **continuously emits the IR light and the IR receiver keeps on checking for the reflected light**. If the light gets reflected back by hitting any object in front of it, the IR receiver receives this light. This way the object or line is detected in the case of the IR sensor.

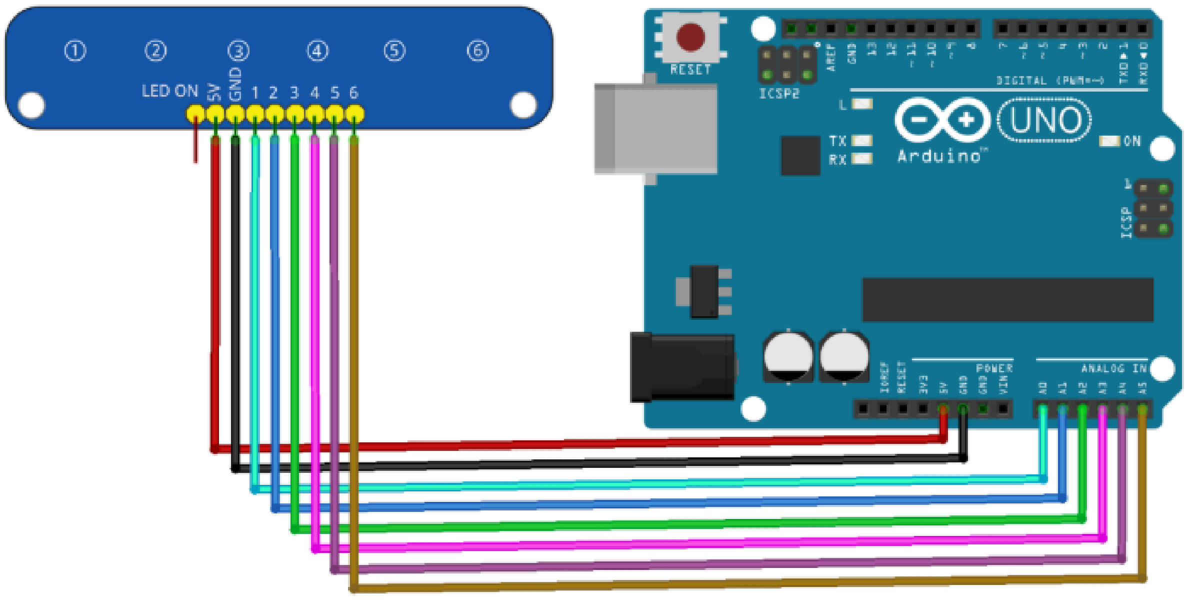


Using the **TCRT500 sensor**, the 6-array IR sensor provides **high voltage for white surfaces** and **low voltage for black surfaces**, ensuring clear signal differentiation. The recommended operational height for the sensor is **3mm–15mm** from the ground.

2. **Setting up the circuit:**

- Connect VCC and GND of the sensor PCB to 5v and GND of Arduino respectively.
- Connect pins 1-6 of the sensor PCB to A0-A5 in proper order.

3. **Circuit diagram:**



4. **Code:**

Arduino Code:

```
#define NUM_SENSORS 6
int sensorPins[NUM_SENSORS] = {A0, A1, A2, A3, A4, A5};

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

void loop() {
  for (int i = 0; i < NUM_SENSORS; i++) {
    int value = analogRead(sensorPins[i]);
    Serial.print("S");
    Serial.print(i + 1);
    Serial.print(": ");
    Serial.print(value);
    Serial.print("\t");
  }
  Serial.println();
  delay(500);
}
```

How this works: *At the hardware level, the IR sensors in the array detect reflected infrared light and generate an analog voltage based on surface color (high for white, low for black). The Arduino Uno reads these voltages using its analog input pins (A0–A5) and converts them to digital values via its 10-bit ADC (0–1023 range). These values are then sent to the serial monitor for real-time tracking. The loop continuously reads sensor data every 500 ms, helping detect and differentiate line positions.*

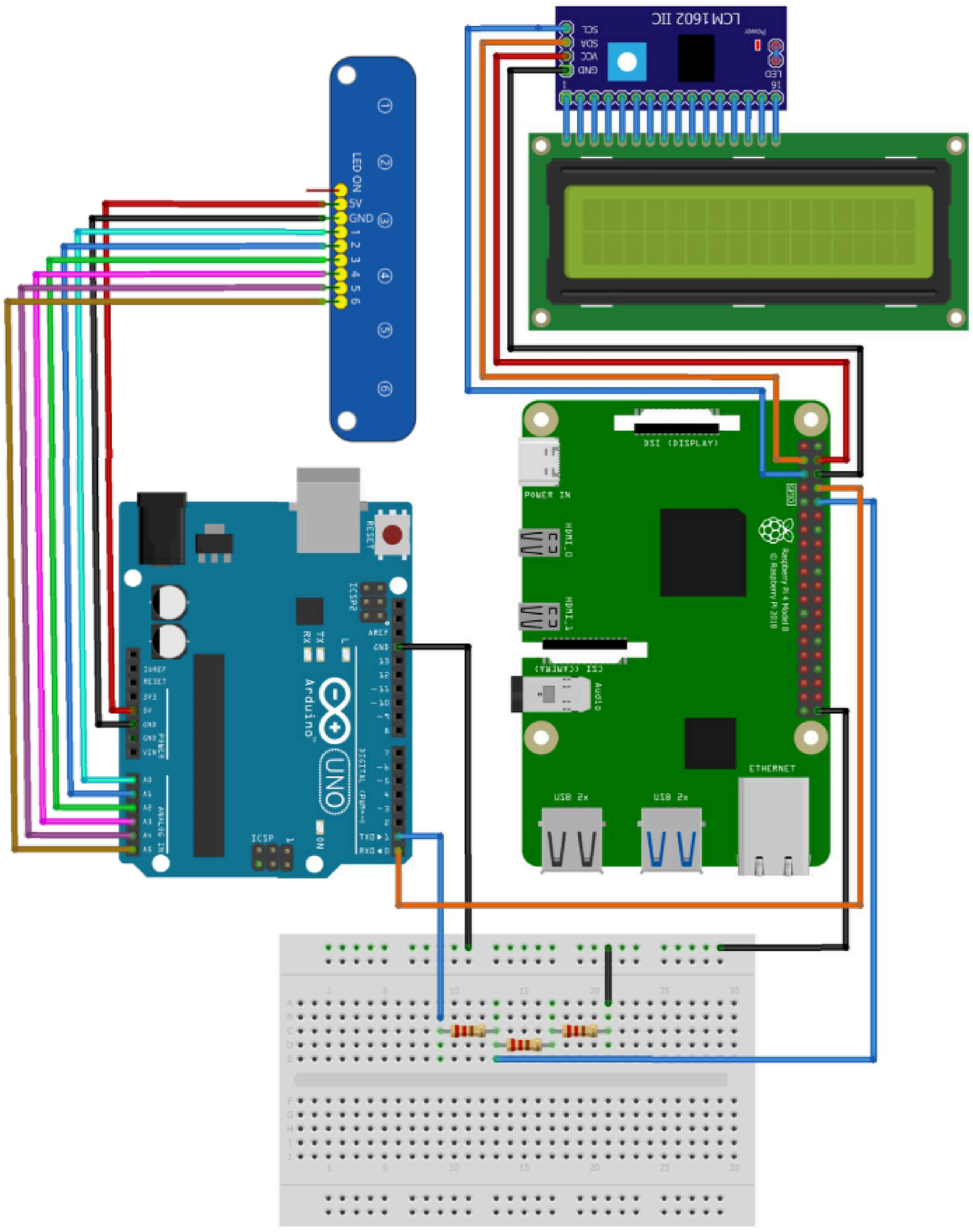
X. Experiment 3 : Interfacing Arduino and Raspberry Pi using UART Protocol to Display IR Sensor Values on an I2C LCD

1. **Setting up the UART connection:**

- Maintain the connections from the previous two experiments without any modifications.
- Connect the TX (Transmit) pin of the Raspberry PI (GPIO 14, pin 8) to the RX (Receive) pin of the Arduino.
- Connect the RX (Receive) pin of the Raspberry PI (GPIO 15, pin 10) to the TX (Transmit) pin of the Arduino through a voltage divider.
- **[IMPORTANT]** Connect the GND pins of both devices together **(Common GND)**

[IMPORTANT] Connect the UART Pins **ONLY** after uploading the code to Arduino.

2. **Circuit diagram:**



3. **Code:**

Arduino Code:

```
#define NUM_SENSORS 6
int sensorPins[NUM_SENSORS] = {A0, A1, A2, A3, A4, A5};
int sensorValues[NUM_SENSORS];

void setup() {
  Serial.begin(9600);
  for (int i = 0; i < NUM_SENSORS; i++) {
    pinMode(sensorPins[i], INPUT);
  }
}

void loop() {
  for (int i = 0; i < NUM_SENSORS; i++) {
    sensorValues[i] = analogRead(sensorPins[i]);
  }

  int threshold = 500;
  int binary[NUM_SENSORS];
  for (int i = 0; i < NUM_SENSORS; i++) {
    binary[i] = (sensorValues[i] > threshold) ? 0 : 1;
  }

  if (binary[0] == 1 && binary[1] == 1 && binary[2] == 1 && binary[3] ==
1 && binary[4] == 1 && binary[5] == 1) {
    Serial.println("STOP");
  }
  else if (binary[0] == 0 && binary[1] == 0 && binary[2] == 0 &&
binary[3] == 0 && binary[4] == 0 && binary[5] == 0) {
    Serial.println("FORWARD");
  }
  else if (binary[0] == 0 || binary[1] == 0) {
    Serial.println("LEFT");
  }
  else if (binary[4] == 0 || binary[5] == 0) {
    Serial.println("RIGHT");
  }
  else {
    Serial.println("UNCERTAIN");
  }

  delay(200);
}
```

Raspberry Pi Code:

```
import serial
from rpi_lcd import LCD
from time import sleep
```

```
# Initialize LCD
lcd = LCD()

# Set up serial connection
ser = serial.Serial('/dev/ttyS0', 9600, timeout=1)

try:
    while True:
        if ser.in_waiting > 0:
            # Read and decode the line from Arduino
            line = ser.readline().decode('utf-8',
errors='ignore').strip()

            # Clear LCD and display the received message
            lcd.clear()
            lcd.text('IR Status: ', 1)
            lcd.text(line, 2)

except KeyboardInterrupt:
    lcd.clear()
    ser.close()
    print("Program terminated")
```

How this works: *The IR sensor array detects reflected infrared light and generates analog voltages based on surface color. The Arduino reads these voltages, converts them to digital values, and determines movement direction (forward, left, right, or stop). The Arduino transmits this decision via UART to the Raspberry Pi. The Raspberry Pi receives the data and updates the I2C LCD display accordingly.*

XI. Lab Task: Connect 3 LEDs with Raspberry Pi and turn them ON/OFF based on the sensor status.

Explanation:

- Left LED turns ON when sensor status is LEFT
- Right LED turns ON when sensor status is RIGHT
- All LEDs turn ON when sensor status is FORWARD
- All LEDs turn OFF otherwise

Deliverables:

- Circuit diagram
- Arduino code
- Demonstration of the working circuit

X. References:

1. Arduino Official Documentation: https://www.arduino.cc/en/Guide
2. Arduino IDE Download: https://www.arduino.cc/en/software
3. Raspberry PI Official Documentation : https://www.raspberrypi.com/documentation/computers/getting-started.html
4. User Manual (6-Array IR Sensor): https://drive.google.com/file/d/1xHp8lToGdmBeAXsS-kn2_UGkGzHITnPw/view