Lab Worksheet 3

CSE461: Introduction to Robotics

Department of Computer Science and Engineering

Lab 03: Interfacing Arduino and Raspberry PI using UART protocol to control an LED with a switch.

I. Topic Overview

This lab introduces the fundamentals of cross-platform microcontroller interfacing using the Arduino Uno and Raspberry PI. Students will learn to establish bidirectional serial communication (UART) between the two devices, enabling them to integrate Arduino's real-time I/O control with the Raspberry PI's computational capabilities. Through hands-on exercises, students will explore voltage-level shifting, GPIO pin functionalities, and protocol-based data exchange. By the end of the lab, students will design a system where a Raspberry PI sends commands to an Arduino to control output devices based on input data, fostering a practical understanding of hybrid embedded system design.

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 cross-platform communication to solve real-world embedded system challenges.
4. Critically troubleshoot hardware and software integration issues.

III. Materials

- Arduino Uno R3
- Raspberry PI 4
- LED (Light Emitting Diode)
- Push-button switch
- Resistor (220Ω for LED)
- 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. Why is this lab important?

In modern embedded systems, multiple microcontrollers often work together to achieve complex tasks. This lab provides hands-on experience in cross-platform communication, an essential skill in robotics and IoT applications. By interfacing an Arduino Uno (optimized for real-time control) with a Raspberry PI (optimized for computation and networking), we will learn how to leverage

the strengths of both platforms. Understanding UART communication enables us to design hybrid embedded systems where microcontrollers exchange data efficiently, paving the way for advanced robotics, automation, and IoT projects.

VI. Setting Up the Arduino IDE in Raspberry PI

Before starting the experiments, ensure the Arduino IDE is properly set up on your Raspberry PI. Follow these steps:

- **Download the Arduino IDE:** Download the latest version of the Arduino IDE from the official website: https://www.arduino.cc/en/software

  Note: Make sure to download the **LINUX (Arm 64 bits)** version of the IDE
- **Install the IDE:** Extract the files and select the **install.sh** file to install the IDE in your system.

  Note: select *Execute in Terminal* during installation
- **Connect the Arduino Board:** Use a USB cable to connect the Arduino board to the computer.
- **Select the Board and Port**:
  - Go to `Tools > Board` and select the appropriate Arduino board (e.g., Arduino Uno).
  - Go to `Tools > Port` and select the COM port to which your Arduino is connected.

This setup ensures that the connected Arduino board is ready for programming and interfacing with external components.

VII. Enabling UART 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.
- Reboot the Raspberry PI:

```
sudo reboot
```

VIII. Experiment: Controlling an LED in Arduino using a Push Button on Raspberry PI

1. **Description the components:**
    - LED (Light Emitting Diode)
    - 2-pin Push-button switch
    - Resistors (220Ω for LED and Voltage Divider)

2. **Setting up the circuit:**

2.1. Raspberry PI 4 Setup:

- Connect one pin of the push button to GPIO 17 (pin 11 on the GPIO header).
- Connect the other pin of the push button to a GND pin.

2.2. Arduino Uno Setup:

- Connect the LED anode (long leg) to a 220Ω resistor and the other end of the resistor to pin 13 on the Arduino.
- Connect the LED cathode (short leg) to a GND pin.

[IMPORTANT] Connect the UART Pins (mentioned in Step 2.3) **ONLY** after uploading the code to Arduino.

Why is this step important: *Arduino's TX/RX pins (0 and 1) are used for both USB uploads and serial communication. If the Raspberry PI is already connected to these pins, it can block the Arduino from receiving new code, causing upload failures. By uploading the code first and then connecting the UART pins, we avoid this conflict and ensure both uploading and serial communication work properly*

2.3. UART Connection Between Raspberry PI and Arduino:

- 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)**

Note: The Raspberry PI uses 3.3V logic, while the Arduino uses 5V logic. To avoid damaging the Raspberry PI, we are using a voltage divider to step down the 5V signal from the Arduino to 3.3V for the Raspberry PI.

3. **Circuit diagram:**

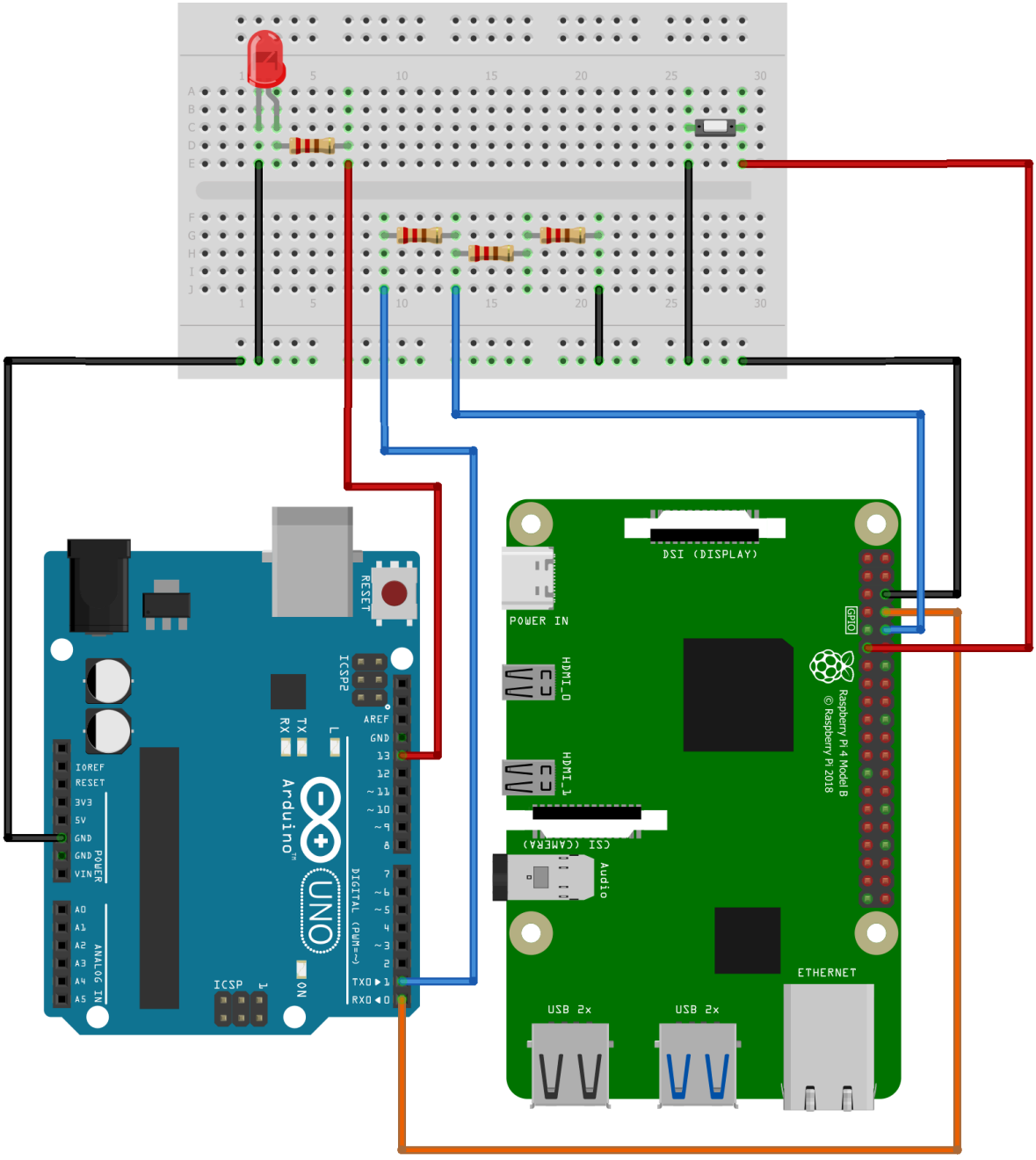


Fig. Arduino and Raspberry PI interfacing

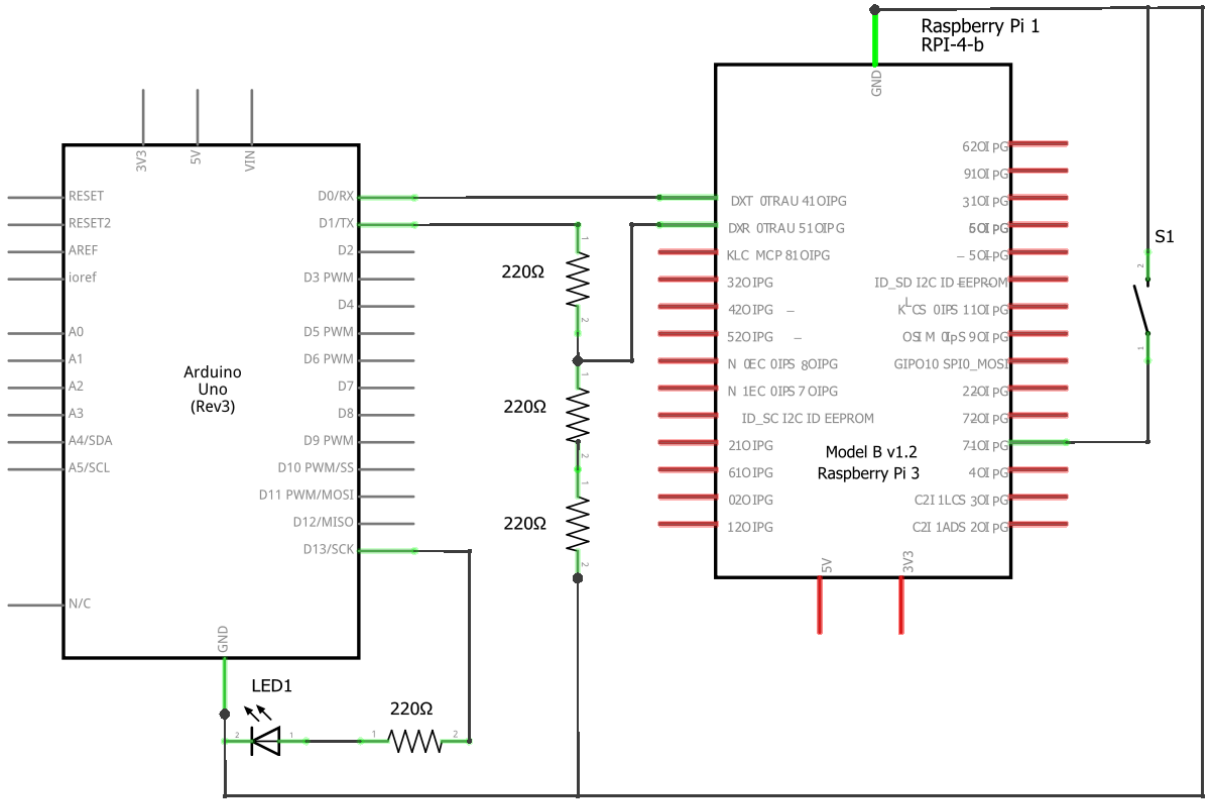
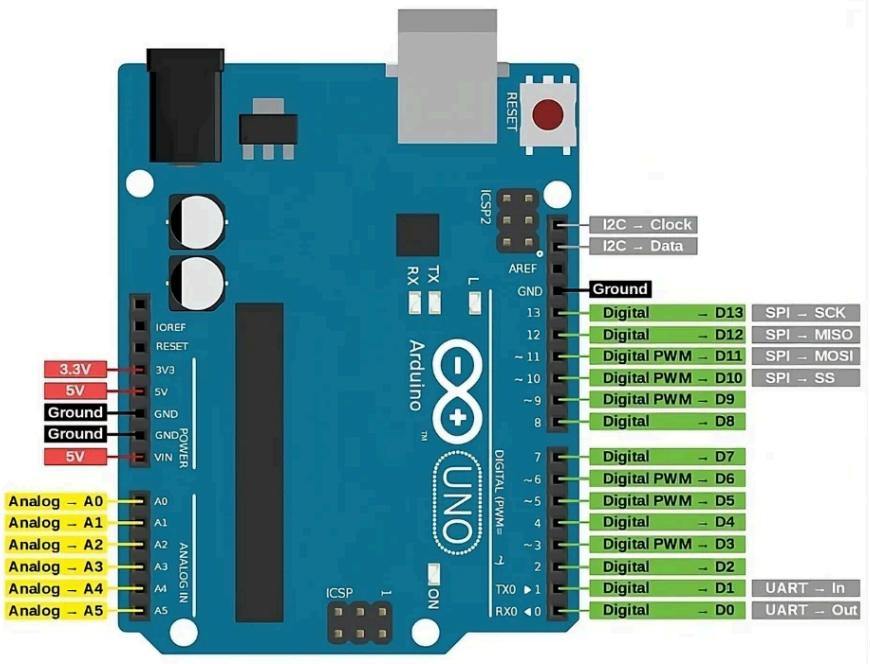
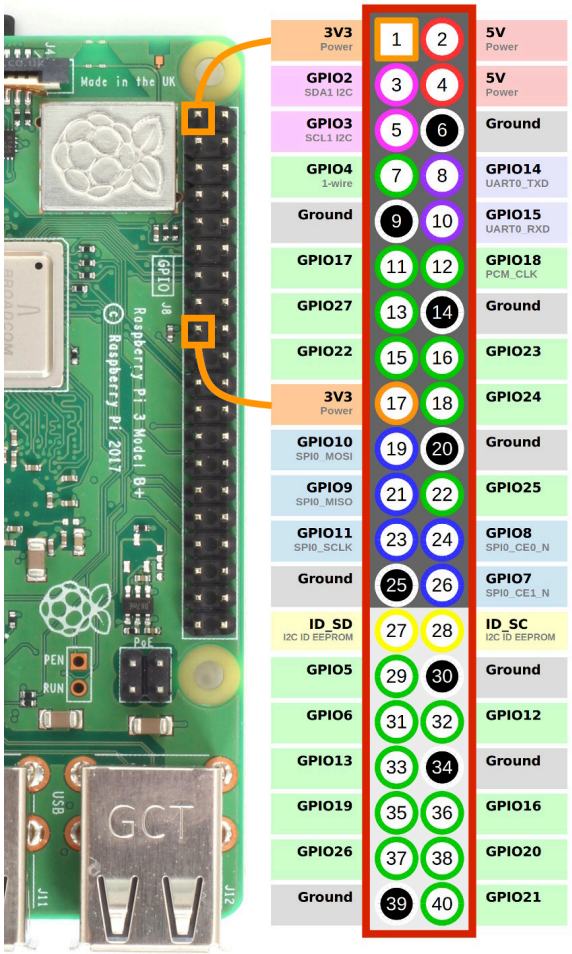


Fig. Arduino and Raspberry PI interfacing Schematic Diagram





4. **Code:**

Raspberry PI Code:

```
import RPi.GPIO as GPIO
import serial
import time

# Set up GPIO for button
button_pin = 17  # GPIO pin 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(button_pin, GPIO.IN)

# Set up UART (Serial) communication with Arduino
ser = serial.Serial('/dev/serial0', 9600)

print("Raspberry Pi is ready. Press the button to blink the LED...")

try:
    while True:
        if GPIO.input(button_pin) == GPIO.LOW:  # If button is pressed
            print("Button pressed! Sending signal to Arduino...")
            ser.write(b'1')  # Send signal to Arduino to blink LED
            time.sleep(0.2)  # Debounce delay
        else:
            print("Button not pressed. Waiting for press...")
        time.sleep(0.1)  # Short delay to avoid excessive CPU usage
except KeyboardInterrupt:
    print("Program interrupted.")
finally:
    GPIO.cleanup()
    print("GPIO cleaned up. Exiting program.")
```

How this works: *At the hardware level, when the button is pressed, it completes a circuit allowing current to flow, pulling GPIO17 to a LOW state (0V). The Raspberry PI detects this state change through its internal pull-up resistor and triggers the code to send a '1' byte over the TX pin (GPIO14, Pin 8) using UART. This signal travels as voltage pulses (3.3V logic) through the TX/RX connection to the Arduino's RX pin (Pin 0). The Arduino's UART hardware module decodes the signal into binary data, processing it in its code to control the LED. Meanwhile, when the button is released, GPIO17 returns HIGH (3.3V), and no signal is sent, keeping the LED unchanged.*

Arduino Code:

```
const int ledPin = 13;  // LED pin

void setup() {
  Serial.begin(9600);  // Initialize UART communication at 9600 baud
rate
  pinMode(ledPin, OUTPUT);  // Set the LED pin as an output
  Serial.println("Arduino is ready. Waiting for command...");
}

void loop() {
  if (Serial.available() > 0) {
    char command = Serial.read();  // Read the incoming byte

    if (command == '1') {
      Serial.println("Button pressed: Blinking LED...");
      digitalWrite(ledPin, HIGH);  // Turn the LED on
      delay(1000);  // Wait for 1 second
      digitalWrite(ledPin, LOW);  // Turn the LED off
      Serial.println("LED blinked.");
    }
  }
}
```

How this works: *the Arduino's UART module continuously monitors its RX pin (Pin 0) for incoming serial data from the Raspberry Pi. When the Raspberry PI sends a '1' byte via UART, the voltage level on the RX pin shifts according to the transmitted data, and the UART hardware converts these pulses into a readable character. The ATmega328P microcontroller then processes this character in the loop() function, and if it matches '1', it sets Pin 13 HIGH (5V), allowing current to flow and lighting up the LED. The LED remains on for 1 second before the microcontroller sets Pin 13 LOW (0V), turning it off. Throughout this process, the Arduino also sends status messages back over TX (Pin 1), which the Raspberry PI can read via its RX pin (GPIO15, Pin 10).*

IX. Lab Task: Complete the experiment shown in class properly. That will serve as the lab task for this week.

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