Cognitive Robotics Lab

Lab 6: Controlling Smart Car Kit from RPi (communication)

Srikar Ganesh 22BAI1080

15-09-2025

Aim:

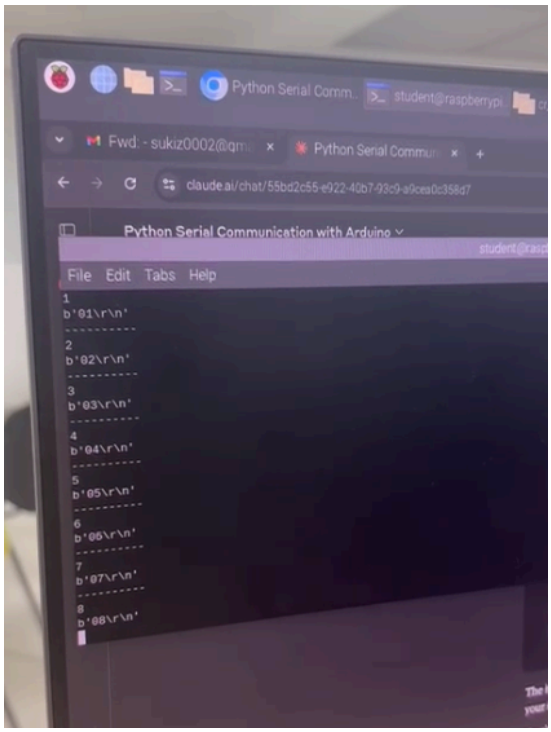
To implement and test serial communication between Raspberry Pi (Rpi4) and Arduino, and to control a Smart Car Kit using Python scripts running on the Raspberry Pi and motor driver code on the Arduino. The four experiments demonstrate:

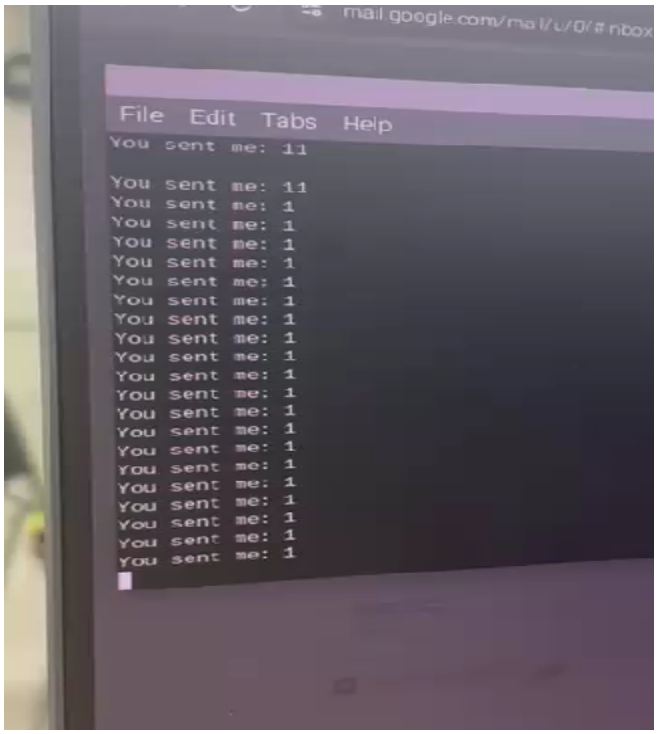
1. Serial testing between Arduino and Raspberry Pi.
2. Controlling Arduino from Raspberry Pi.
3. Forward movement of the Smart Car.
4. Speed control of the Smart Car.

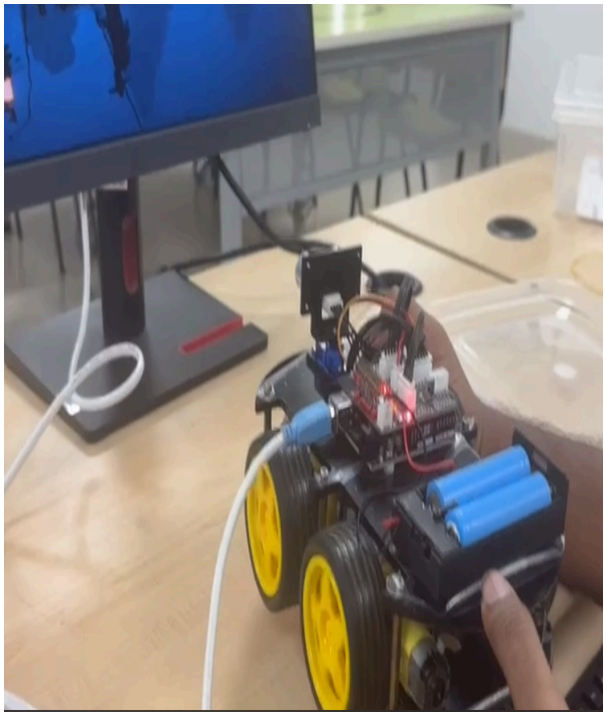
Components:

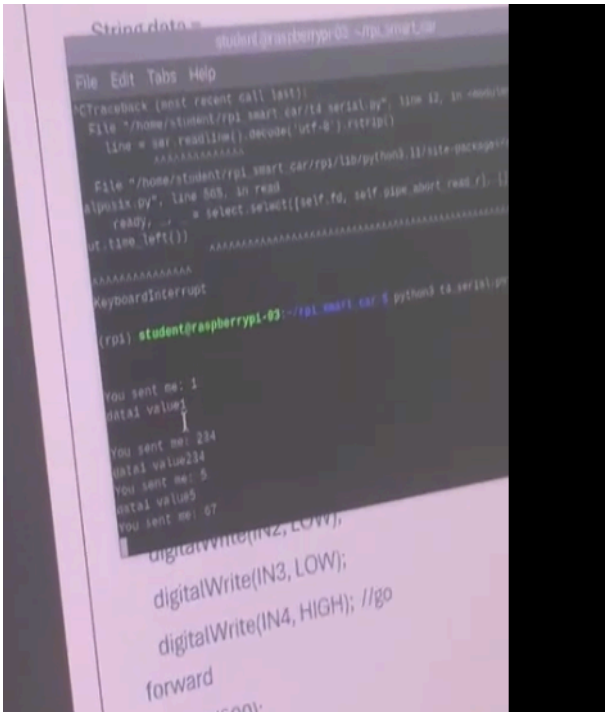
1. Raspberry Pi 4 (with Raspbian OS installed and Python support)
2. Arduino Uno / Mega (or compatible board)
3. Smart Car Kit (including chassis, wheels, and DC motors)
4. Motor Driver IC (L298N or similar) for motor control
5. Jumper wires and connectors for circuit connections
6. USB cable (Arduino ↔ Raspberry Pi communication)
7. Breadboard (optional, for wiring convenience)
8. Power supply (battery pack or regulated adapter for Smart Car Kit)
9. Arduino IDE (to upload .ino code)
10. Python 3 with pyserial library installed on Raspberry Pi

Output:









Source Code:

```
============================
EXPERIMENT 1 — Serial testing between Arduino and RPi
============================

--- serial_test.ino ---
char dataString[50] = {0};
int a =0;

void setup() {
  Serial.begin(9600);             //Starting serial communication
}

void loop() {
  if(a<10)
    {
      a++;                        // a value increase every loop
      sprintf(dataString,"%02X",a); // convert a value to hexa
      Serial.println(dataString);   // send the data
      delay(1000);
    }// give the loop some break
}
```

--- serial_test.py ---

```
import serial

ser = serial.Serial('/dev/ttyACM0',9600)
s = [0]
while True:
 read_serial=ser.readline()
 s[0] = str(int (ser.readline(),16))
 print(s[0])
 print(read_serial)
```

==========

EXPERIMENT 2 — Controlling Arduino from RPi

==========

--- controlfromRpi4.ino ---

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

void loop() {
  if (Serial.available() > 0) {
    String data = Serial.readStringUntil('\n');
    Serial.print("You sent me: ");
    Serial.println(data);
  }
}
```

--- controltoArduino.py ---

```
#!/usr/bin/env python3
import serial
import time

if __name__ == '__main__':
    ser = serial.Serial('/dev/ttyACM1', 9600, timeout=1)
    ser.reset_input_buffer()

    while True:
        line='1'
        line_encode=line.encode()
        ser.write(line_encode);
        #line = ser.readline().decode('utf-8').rstrip()
        #print(line)
        time.sleep(1)
```

```
============================
EXPERIMENT 3 — Forward movement of Smart Car
============================

--- ForwardMovement.ino ---
#define ENA 5
#define ENB 6
#define IN1 7
#define IN2 8
#define IN3 9
#define IN4 11
int data1=0;

void setup() {
 pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  if (Serial.available() > 0) {
    String data = Serial.readStringUntil('\n');
    Serial.print("You sent me: ");
    Serial.println(data);
    data1=data.toInt();
    //data1 = data1+2;
    Serial.print("data1 value");
    Serial.println(data1);
    digitalWrite(ENA, data1);
    digitalWrite(ENB, data1);
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);  //go forward
    delay(500);
  }
}

--- ForwardMovement.py ---
#!/usr/bin/env python3
import serial
```

```
import time

if __name__ == '__main__':
    ser = serial.Serial('/dev/ttyACM4', 9600, timeout=1)
    ser.reset_input_buffer()

    while True:
        line='1'
        line_encode=line.encode()
        ser.write(line_encode);
        #line = ser.readline().decode('utf-8').rstrip()
        #print(line)
        time.sleep(1)


============================
EXPERIMENT 4 — Speed control of Smart Car
============================

--- Acce.ino ---
#define ENA 5
#define ENB 6
#define IN1 7
#define IN2 8
#define IN3 9
#define IN4 11
int data1=0;

void setup() {
 pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  Serial.begin(9600);

}

void loop() {
    if (Serial.available() > 0)
    {
    String data = Serial.readStringUntil('\n');
    Serial.print("You sent me: ");
    Serial.println(data);
    data1=data.toInt();
```

```
    //data1 = data1+2;
    Serial.print("data1 value");
    Serial.println(data1);
    digitalWrite(ENA, data1);
    digitalWrite(ENB, data1);
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);  //go forward
    delay(500);
    }
}

--- AccelerateSpeed.py ---
#!/usr/bin/env python3
import serial
import time
i=0
if __name__ == '__main__':
    ser = serial.Serial('/dev/ttyACM4', 9600, timeout=1)
    ser.reset_input_buffer()

while True:
    line=str(i)
    line_encode=line.encode()
    ser.write(line_encode);
    line = ser.readline().decode('utf-8').rstrip()
    print(line)
    time.sleep(1)

    i=i+1
```

Explanation:

In these experiments, communication between the Raspberry Pi and Arduino was established using serial protocols. The Arduino acted as the controller for the motors in the Smart Car Kit, while the Raspberry Pi served as the high-level interface that transmitted commands through Python scripts. Each task gradually increased in complexity: beginning with basic serial data transmission and reception, progressing to sending control signals from the Pi to the Arduino, and finally extending to motor control for forward movement and variable speed operation. This

step-wise design ensured that both hardware and software communication layers were verified before integrating motor functions.

The Arduino programs were written in C++ (.ino) and uploaded through the Arduino IDE, while the Raspberry Pi scripts were written in Python and executed with the pyserial library. When the Python code sent characters or numbers, the Arduino read these signals via its serial buffer, interpreted them as integers, and used them to set motor speed and direction using digital outputs and PWM on the L298N motor driver. This design effectively demonstrated how higher-level devices like the Raspberry Pi can offload real-time motor control tasks to microcontrollers like Arduino, creating a reliable and modular control architecture.

Result:

The experiments successfully demonstrated serial communication between Raspberry Pi and Arduino, enabling remote control of a Smart Car Kit. Commands sent from the Raspberry Pi were correctly received and executed by the Arduino, resulting in visible forward movement of the car and gradual acceleration when speed control was tested. This confirms that the system design was effective in achieving the stated aim of integrating Raspberry Pi–Arduino communication for robotic vehicle control.