SRM Institute of Science and Technology

College of Engineering and Technology

Department of Electronics and Communication Engineering

18ECO109J-Embedded System Design Using Raspberry Pi
2022-23 (Even Semester)

Mini Project Report

Name	: Rithika Varaganti , Madavaram Reddy Vinathi
Register No.	: RA2011026010130, RA2011030010133
Day / Session	: 1/ P-3,4
Venue	: TP-RF LAB
Project Title	: Resistive Sensors
Lab Supervisor	: SUGANTHI BRINDHA G
Team Members	: 1) Rithika Varaganti **(**RA2011026010130**)**
	2) Madavaram Reddy Vinathi (RA2011030010133)

Particulars	**Max. Marks**	**Marks Obtained**
Objective & Description	05	
Algorithm,Flowchart,Program	20	
Demo verification	10	
Viva	10	
Report	05	
Total	**50**	

REPORT VERIFICATION

Date	:
Staff Name	: SUGANTHI BRINDHA G

RESISTIVE SENSORS

OBJECTIVE:

To connect a variable resistor to a Raspberry Pi and measure the position of its rotation.

ABSTRACT:

In this project, we explore the use of a variable resistor with a Raspberry Pi and measure the position of its rotation. A variable resistor, also known as a potentiometer, is a three-terminal electronic component that allows us to vary the resistance between its terminals by rotating a knob or slider. By connecting a variable resistor to an analog input pin on the Raspberry Pi, we can read the voltage level corresponding to the position of its rotation using a software library or programming language.

To measure the position of the variable resistor's rotation accurately, we need to calibrate the readings by mapping the voltage values to the corresponding rotation angles. This can be achieved by connecting a voltmeter to the variable resistor and rotating it to different positions while measuring the corresponding voltage values. Once we have obtained a set of voltage-angle pairs, we can use a suitable curve-fitting algorithm to interpolate the rotation angle corresponding to a given voltage reading.

With this setup, we can create a wide range of applications, such as controlling the brightness of an LED or adjusting the speed of a motor. By using a variable resistor as a user interface, we can provide a more intuitive and flexible control over these applications.

HARDWARE / SOFTWARE REQUIRED

- Raspberry Pi
- M/F jumper wires
- Breadboard
- Capacitor
- Two resistors
- Trimpot

BLOCK DIAGRAM/CONNECTION DIAGRAM

You can measure resistance on a Raspberry Pi using nothing more than a capacitor, a couple of resistors, and two GPIO pins. In this case, you will be able to read the position of the knob on a small variable resistor (trimpot).

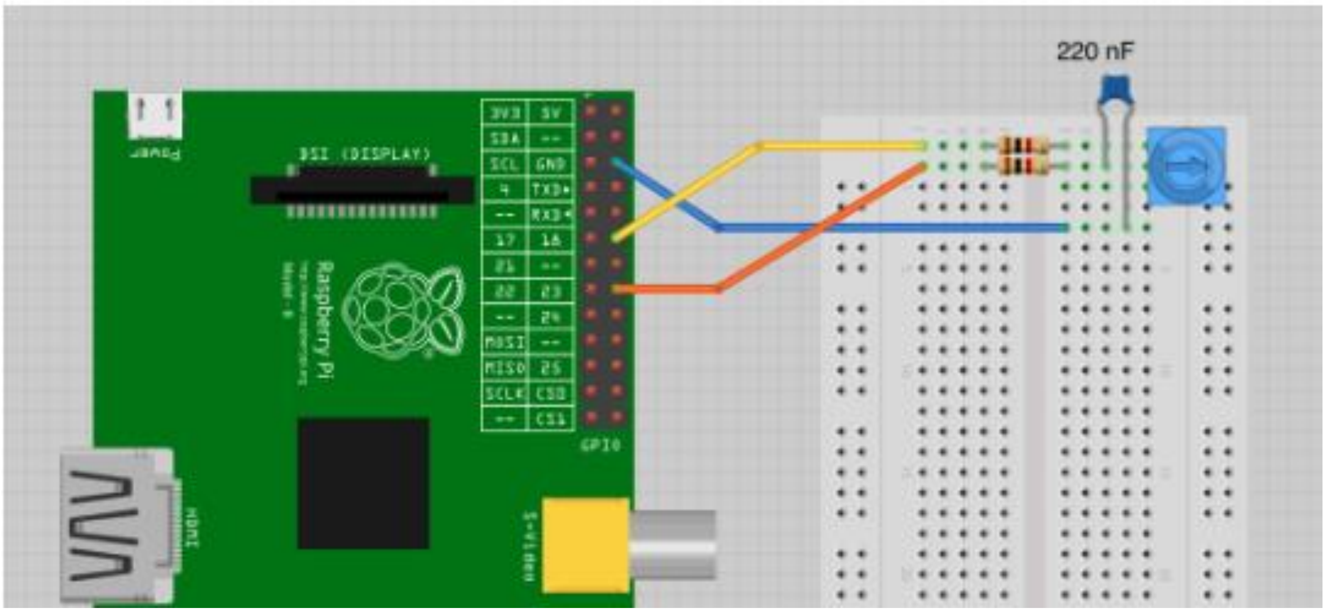
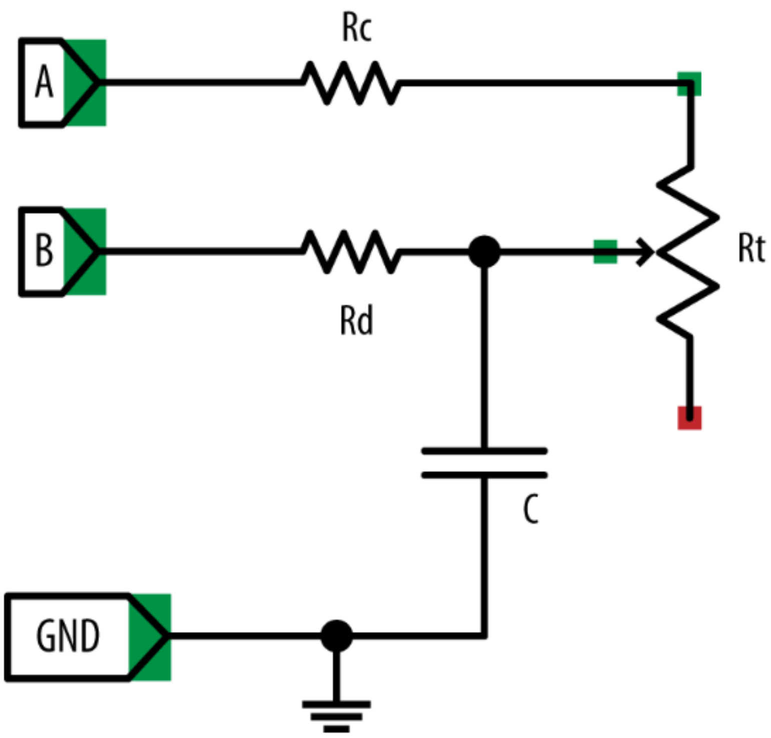


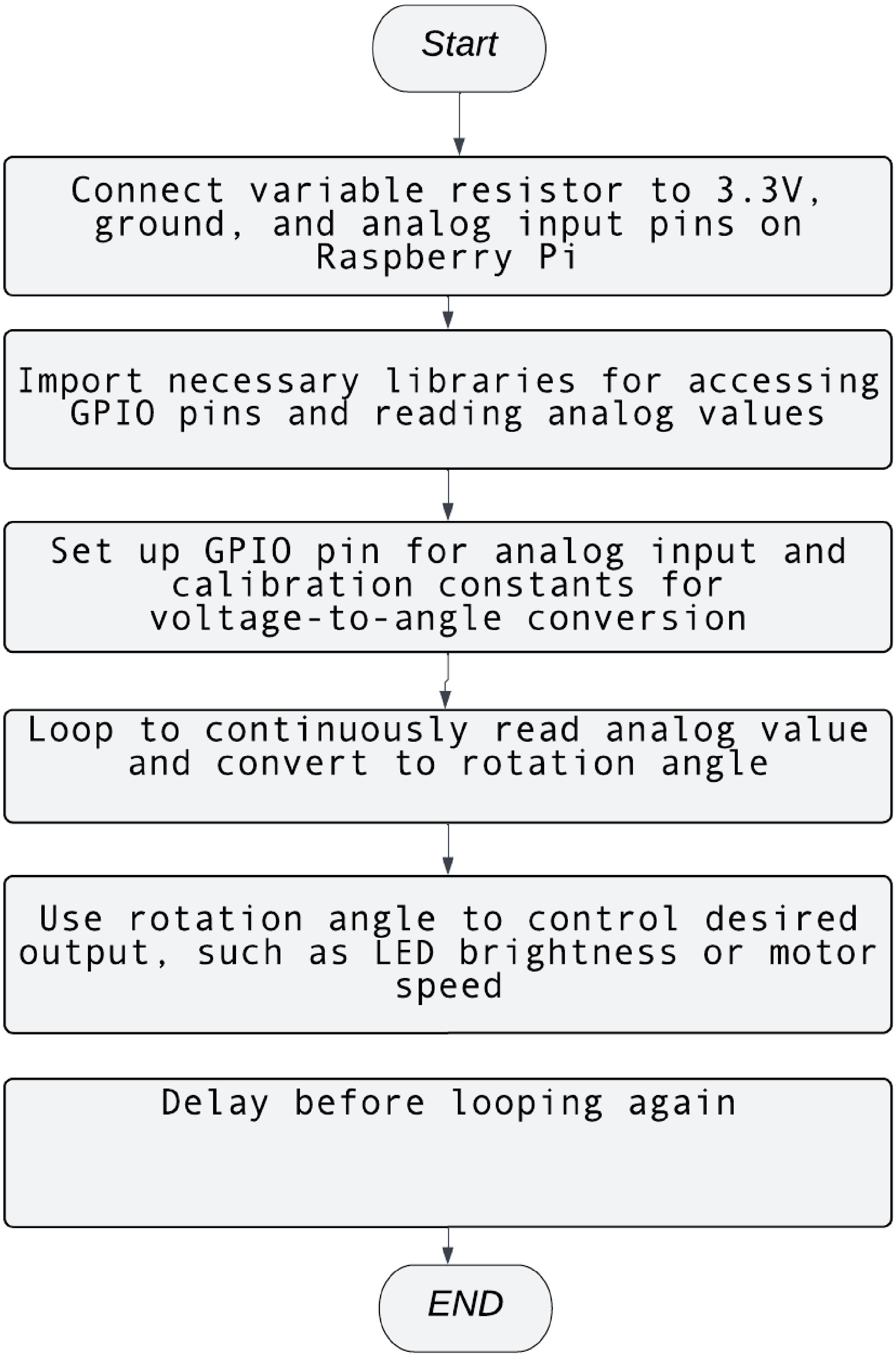
Figure 8-2. Measuring resistance using step response



ALGORITHM:

1) Connect one end of the variable resistor to a 3.3V pin on the Raspberry Pi.
2) Connect the other end of the variable resistor to a ground pin on the Raspberry Pi.
3) Connect the middle terminal of the variable resistor to an analog input pin on the Raspberry Pi.
4) Open a terminal window or IDE on the Raspberry Pi and import the necessary libraries for accessing the GPIO pins and reading analog values.
5) Set up the GPIO pin for the analog input by setting it to input mode.
6) Use a loop to continuously read the analog value from the analog input pin.
7) Convert the analog value to a voltage reading using the appropriate conversion formula.
8) Calibrate the voltage reading to the corresponding rotation angle using a curve-fitting algorithm or a lookup table.
9) Use the rotation angle to control the desired output, such as the brightness of an LED or the speed of a motor.

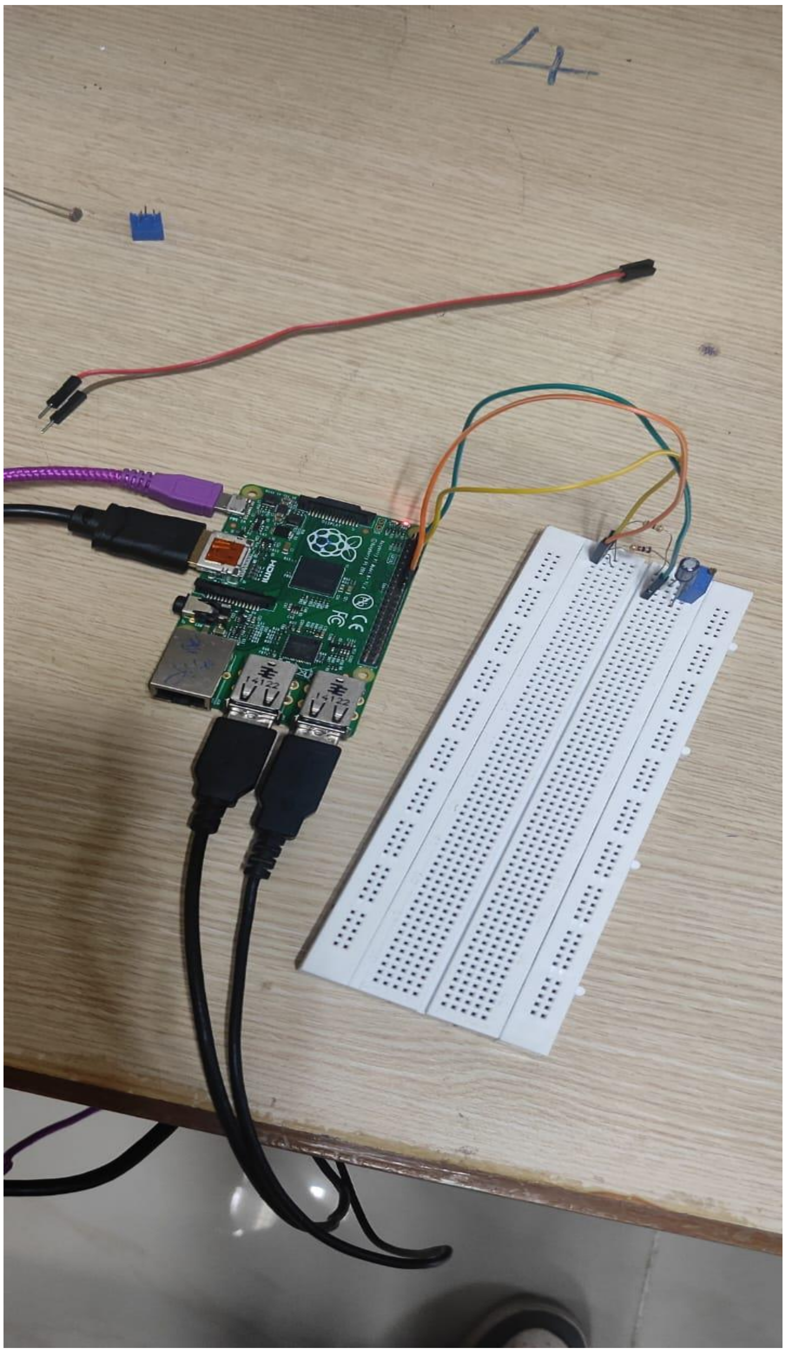
FLOW CHART:

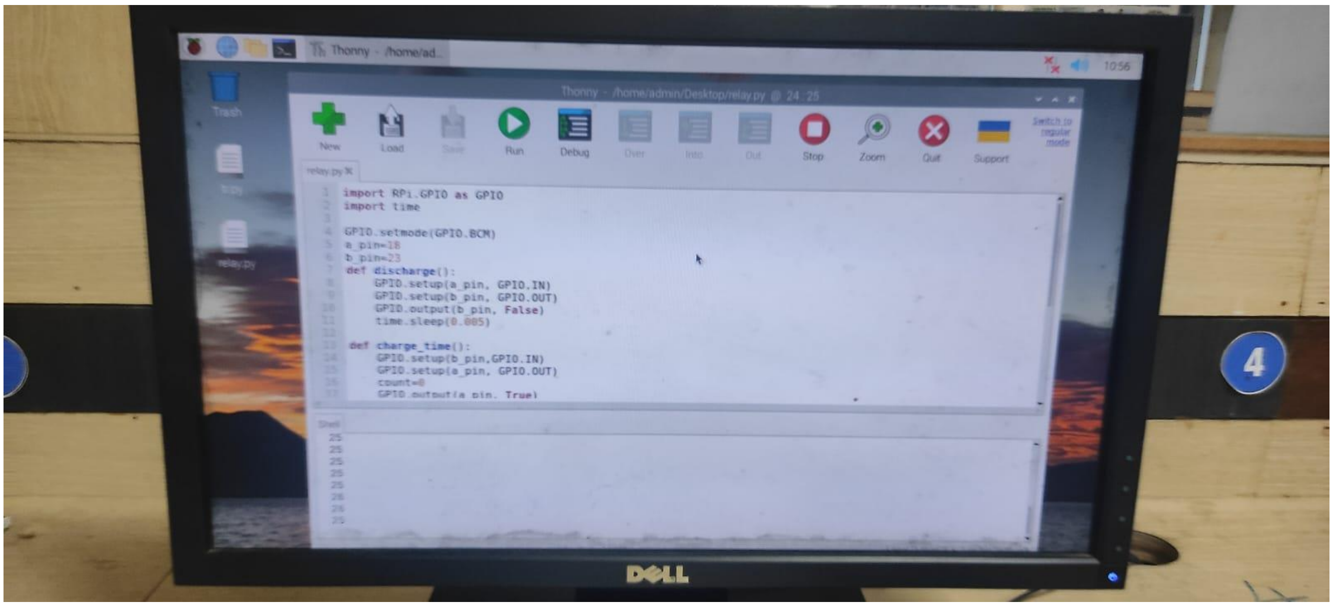

Start
Connect variable resistor to 3.3V, ground, and analog input pins on Raspberry Pi
Import necessary libraries for accessing GPIO pins and reading analog values
Set up GPIO pin for analog input and calibration constants for voltage-to-angle conversion
Loop to continuously read analog value and convert to rotation angle
Use rotation angle to control desired output, such as LED brightness or motor speed
Delay before looping again
END

PROGRAM:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

a_pin = 18
b_pin = 23

def discharge():
    GPIO.setup(a_pin, GPIO.IN)
    GPIO.setup(b_pin, GPIO.OUT)
    GPIO.output(b_pin, False)
    time.sleep(0.005)

def charge_time():
    GPIO.setup(b_pin, GPIO.IN)
    GPIO.setup(a_pin, GPIO.OUT)
    count = 0
    GPIO.output(a_pin, True)
    while not GPIO.input(b_pin):
        count = count + 1
    return count

def analog_read():
    discharge()
    return charge_time()

while True:
    print(analog_read())
    time.sleep(1)
```

OUTPUT:





```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
a_pin=18
b_pin=23
def discharge():
     GPIO.setup(a_pin, GPIO.IN)
     GPIO.setup(b_pin, GPIO.OUT)
     GPIO.output(b_pin, False)
     time.sleep(0.005)

 def charge_time():
     GPIO.setup(b_pin,GPIO.IN)
     GPIO.setup(a_pin, GPIO.OUT)
     count=0
     GPIO.output(a_pin, True)
```

```
25
25
25
25
25
26
26
26
25
```

DEMONSTARTION PICTURES:

Video Link:

https://drive.google.com/file/d/1LWQG0M370Go_E54TIhYUfd-j97xJi2U7/view?usp=share_link

REAL TIME CONSTRAINTS:

The real-time constraints of measuring the position of a variable resistor with a Raspberry Pi depend on various factors such as the sampling rate, the accuracy of the analog-to-digital converter (ADC). If we change the fixed resistor value to 1kΩ, the output voltage from the voltage divider circuit will decrease. Suppose we use a 10nF capacitor in parallel with the humidity sensor. If we change the capacitor value to 1nF, the time constant of the RC circuit decreases.

CONCLUSION:

In conclusion, connecting a variable resistor to a Raspberry Pi and measuring its rotation can be a useful project for controlling various outputs, such as LEDs or motors. To achieve accurate and real-time measurements, it is important to consider factors such as the sampling rate, ADC accuracy, software complexity, and interference. By carefully optimizing these factors, it is possible to obtain precise and timely measurements of the variable resistor's position, allowing for precise control of the desired output.

References:

1) https://www.jeremymorgan.com/tutorials/raspberry-pi/how-to-read-analog-inputs-for-raspberry-pi-using-the-mcp3008/
2) https://www.kevindarrah.com/wiki/index.php?title=Raspberry_Pi_Analog_Input_Using_an_MCP3008
3) https://learn.adafruit.com/raspberry-pi-analog-to-digital-converters/overview