ANT: PH Sensing And Peristaltic Pumps

By Raymond Duong, ANT Computer Engineer

Last Updated: November 11, 2021

In this report, the Grove PH sensor,
Waveshare's High Precision AD/DA Board, and
peristaltic pumps will be tested together
for ANT's PH balance feature.

Note: All tests will be conducted at room temperature
so the following results are most accurate in the temperature
range (18°C to 25°C).

Table of Contents

Introduction

Why is PH balance automation necessary for the ANT system?

PH balance automation is a crucial feature of the ANT system. Plants need a specific PH range to grow since PH level too high or too low affects nutrient availability and other factors important to plant health. Ex. PH levels being too acidic can burn the roots or even be toxic to plants. For our ANT system, we want to maintain this PH range (PH 5.5 to PH 7.0) so plants in our system can grow in a nice stable environment and automation of this balance reduces the maintenance needed.

How would PH automation work in the ANT system?

The PH sensor in our system will frequently send back read PH values measured from the water supply to the main control system (Raspberry PI) and those values will be processed. If the PH value readings are below or above the desired PH range, one of two peristaltic pumps will be activated to dose PH UP or PH DOWN into the water supply. PH UP will be dosed if the PH value read is below the desired range and in the opposite of being above range, PH DOWN will be dosed.

PH Sensing and Pumps With Mechanical Relays

Goal:

Use Raspberry PI and relay module to turn on peralistic pump when ph sensor reading returns a value out of desired set range.

Materials:

Raspberry PI 3b+

Waveshare High Precision AD/DA Board

Grove PH sensor

Relay Module

Jumper Cables

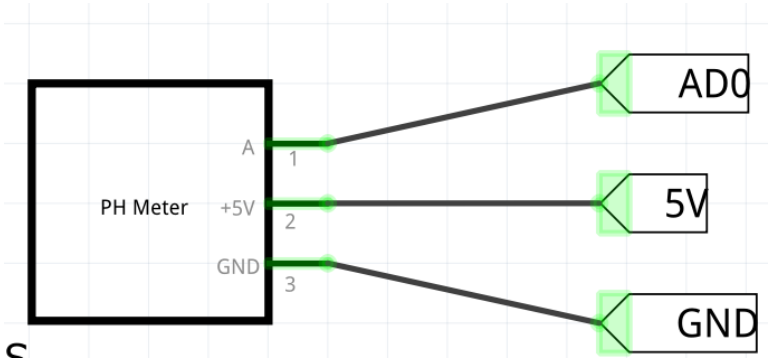
Peristaltic Pump

1N4007 Diode

Motor Speed Controller

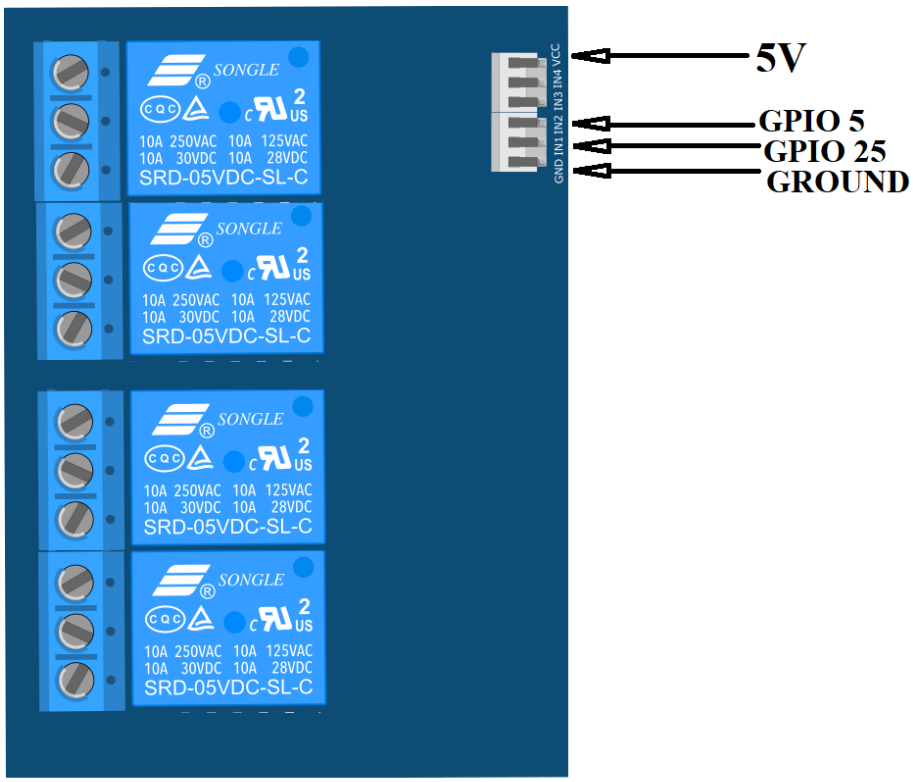
Schematic:

Raspberry PI and PH sensor:

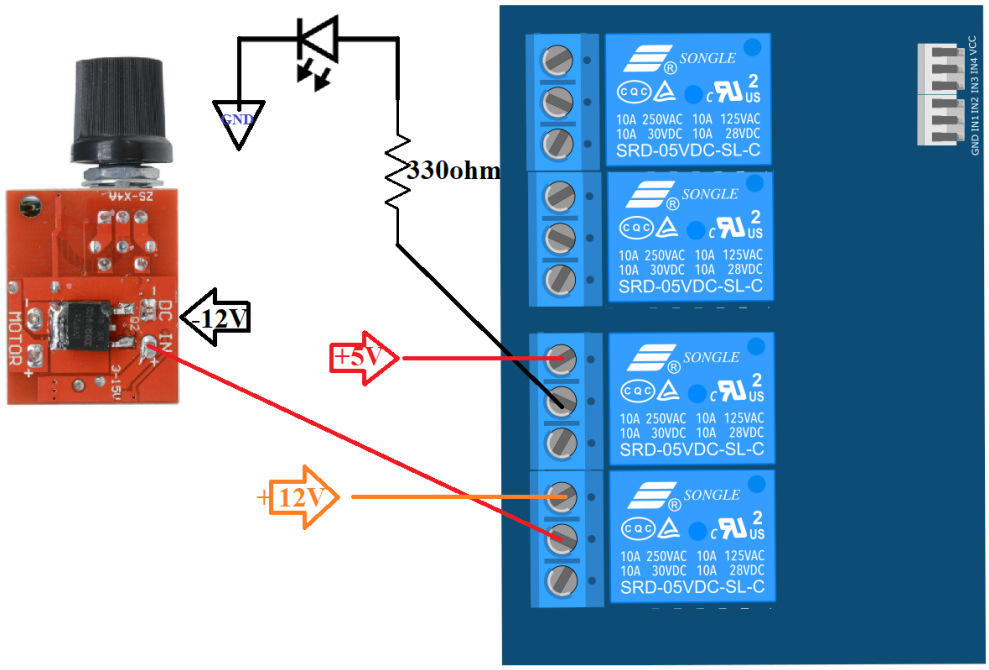


Schematic for Waveshare AD/DA board and ph sensor connection

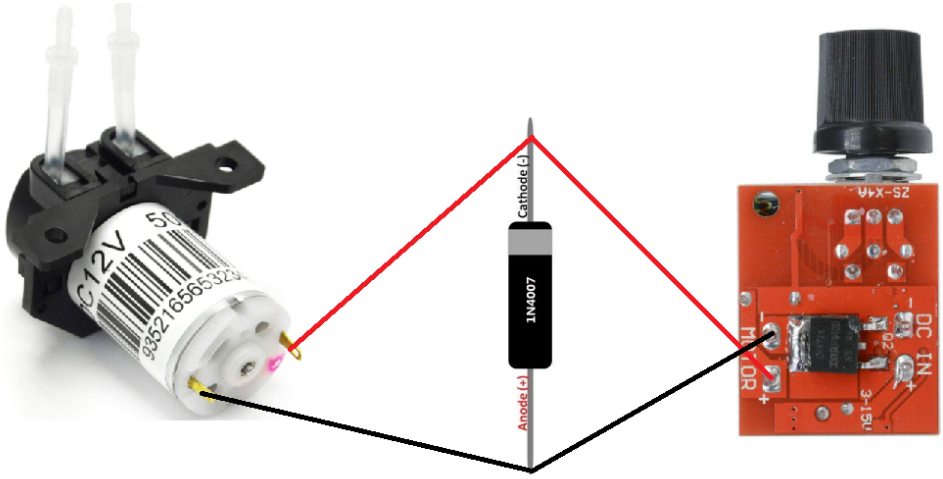
Raspberry PI, Relay module, and pumps:



Connection between Raspberry PI and relay module

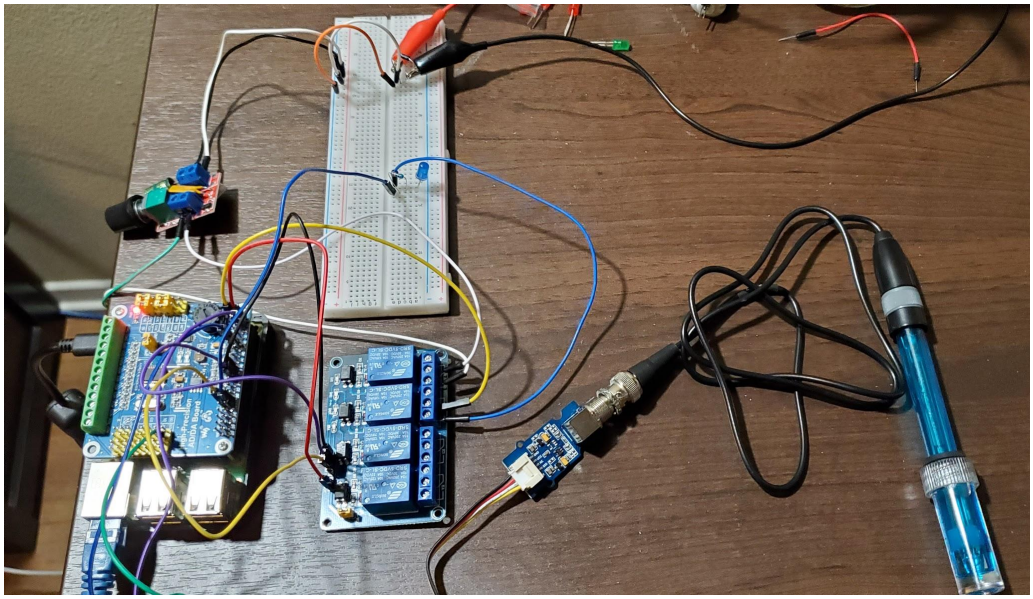


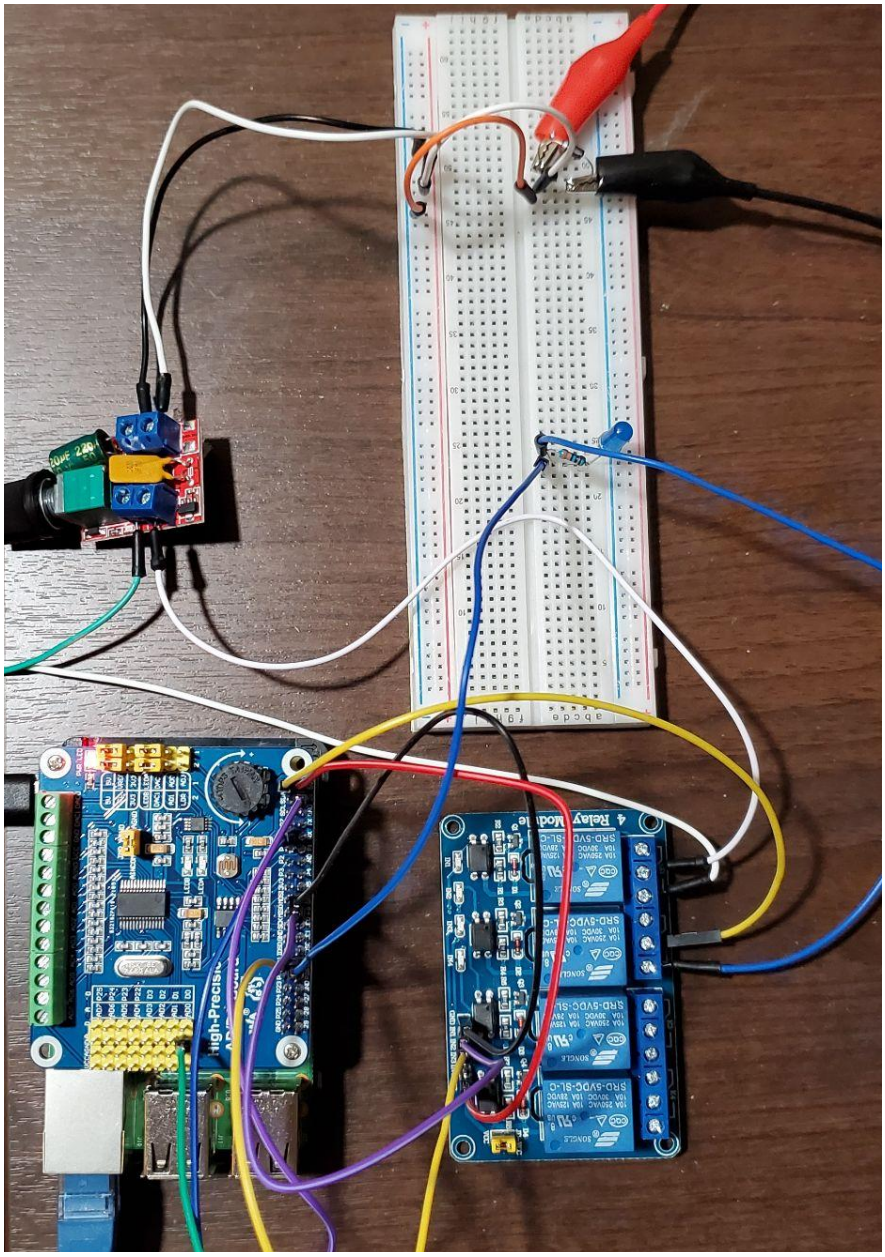
Connection between DC motor speed controller and relay module.



Connection between DC motor speed controller and pump

Final Connection Result:





Procedure:

1. Connect Raspberry PI, relay module, motor speed controller, and pumps as shown in the connection diagram above. (Note: Another pump and motor speed controller can be connected in place of the led and resistor above. Connect the pump in a similar fashion to the first.)
2. Setup ph sensor as shown in “ANT:PH Sensor Testing Report” in section PH Sensor:Raspberry PI and Waveshare High Precision AD/DA Board.”
3. Create a new python file, ph_automation.py, in the directory of phsensor.py and other files shown in “ANT:PH Sensor Testing Report.”

4. Copy the following code into ph_automation.py:

```
import RPi.GPIO as GPIO
import phsensor
import sys
import time



PH_UP = 25
PH_DOWN = 5

PH_MIN = 5.5
PH_MAX = 7
```

```
def GPIOSetup():
    GPIO.setup(PH_UP, GPIO.OUT, initial = GPIO.HIGH)
    GPIO.setup(PH_DOWN, GPIO.OUT, initial = GPIO.HIGH)
if __name__ == '__main__':
    try:
        GPIO.setmode(GPIO.BCM)
        GPIOSetup()
        ph_sensor = phsensor.PHSensor(0,14) #phsensor object
        while True:
            #ph sensing from ad da board
            ph = round(ph_sensor.read_ph(0),1)

            #if ph level is below range
            if ph < PH_MIN:
                #turn on PH up pump
                GPIO.output(PH_UP, GPIO.LOW)
                print(f"PH UP pump on at PH:{ph}")
                #time.sleep(pump duration)
                time.sleep(1)
                #turn off PH up pump
                GPIO.output(PH_UP, GPIO.HIGH)
                print("PH UP pump off")

            #if ph level is above range
            if ph > PH_MAX:
                #turn on PH down pump
                GPIO.output(PH_DOWN, GPIO.LOW)
                print(f"PH DOWN pump on at PH:{ph}")
                #time.sleep(pump duration)
                time.sleep(1)
                #turn off PH down pump
                GPIO.output(PH_DOWN, GPIO.HIGH)
                print("PH DOWN pump off")

            #ph sensing interval
            time.sleep(60)
    except KeyboardInterrupt:
        GPIO.cleanup()
        sys.exit('PH balancing ended')
```

5. Run ph_automation.py. (Note: If needed ‘PH_MIN’ and ‘PH_MAX’ can be changed to desire ph range).
6. Dip the probe in solution within set range. No pumps or leds should turn on. Then, test with various solutions outside set range. Note down which pump/led turns on when the PH probe is set in certain solutions.

Results:

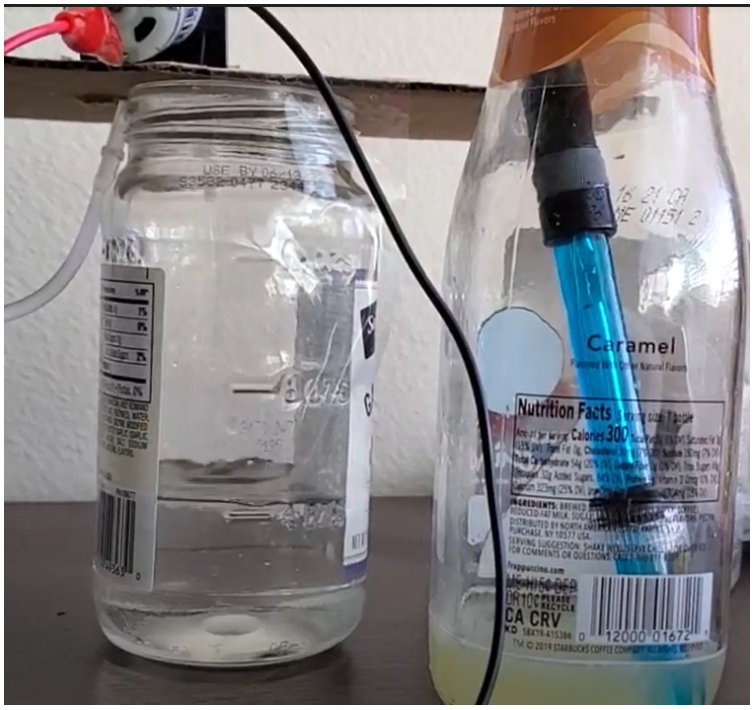
Test Case 1: PH probe in tap water (PH 6 ~ 7)

As expected no devices became active during this test with tap water. Tap water’s expected PH is within desired set PH range 5.5 to 7.0, so there is no need to turn on pumps for PH UP or PH DOWN.

```
pi@raspberrypi:~/Code/ANT-Automated-Nutrition-Technique/Sensors/PH/Waveshare_AD $ python3 ph_automation.py
ID Read success
```

Terminal Feedback With No Print of Pumps Turning ON

Test Case 2: PH probe in lemon juice (2 ~ 3)



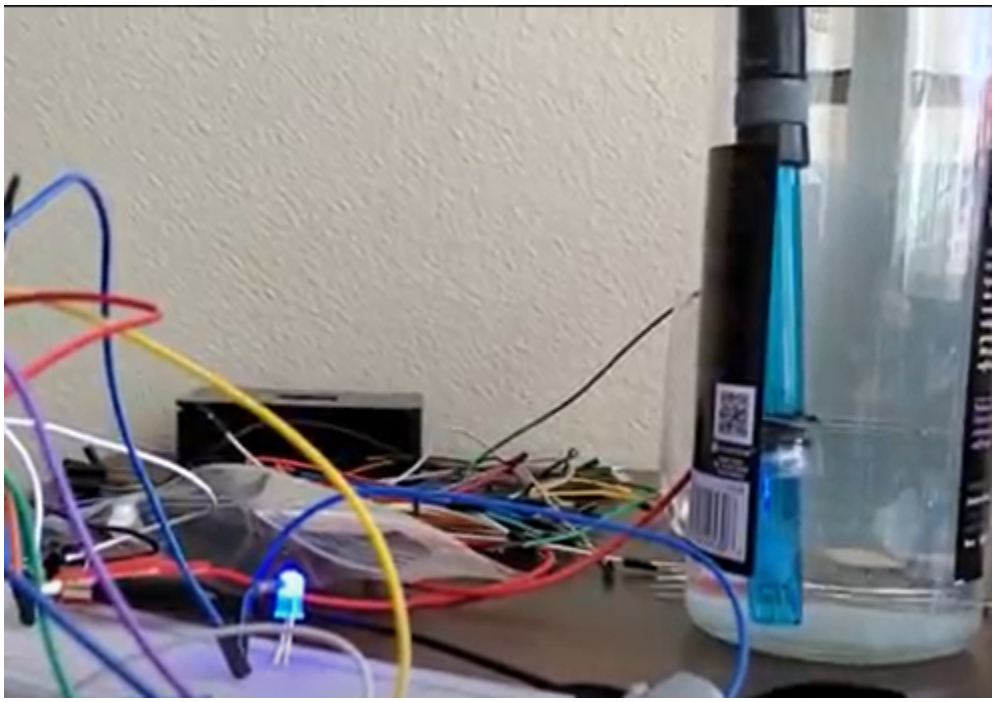
Peristaltic turned on when probe dipped in lemon juice

The expected PH of lemon juice is between PH 2 and PH 3. This means the PH level of lemon juice is below our set range of PH 5.5 to PH 7.0. PH UP should be dosed and in our test, the peristaltic pump turned on when the ph probe is submerged in lemon juice. PH UP was dosed so our expectations were reached.

```
PH DOWN pump on at PH:7.2
PH DOWN pump off
PH DOWN pump on at PH:7.3
PH DOWN pump off
PH DOWN pump on at PH:7.4
PH DOWN pump off
PH UP pump on at PH:2.3
PH UP pump off
PH UP pump on at PH:2.3
PH UP pump off
PH UP pump on at PH:2.3
PH UP pump off
```

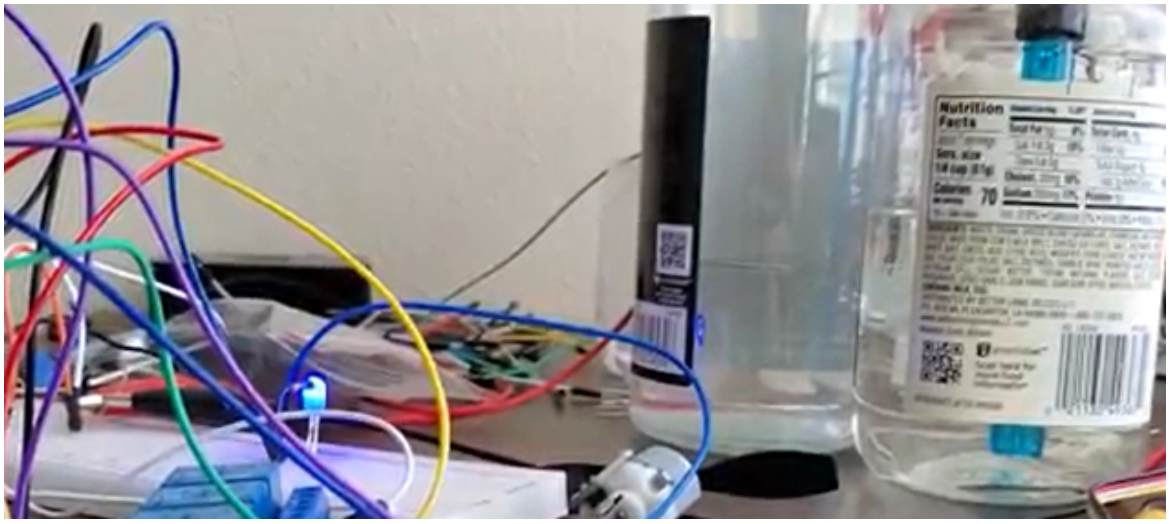
Terminal Feedback from peristaltic pump turning on

Test Case 3: PH probe in baking soda (~ 8.3)



Blue LED turns on when PH probe dipped in baking soda solution

When the PH probe is dipped in baking soda solution, the blue LED turns on as expected. Baking soda solution expected PH is around 8.3 which is above or desired PH range of PH 5.5 to PH 7.0, so PH DOWN should be dosed. Since in this test, the PH DOWN pump is represented by the blue LED, PH DOWN was dosed as expected.



Blue LED still turns on when place from baking soda solution back to tap water

One concern during this test was that the blue LED still turns on when the PH probe is transferred from the baking soda solution to the normal tap. This is unexpected because tap water PH is within our set range and no devices should turn on as shown in our first test with tap water.

After around 1 minute, the blue LED stopped turning on and when running main2.py to see our PH level, the PH value returned back to our expected between PH 6 and PH 7. The initial problem arises when the PH probe is suddenly switched into another solution within our set PH range again. The PH probe needed time to stabilize to the PH of the new solution.

In the case of the ANT system, this delay should not cause a problem to our system since the PH probe will be set in one spot in our system and no sudden change of solution should occur often when our system is operating. A fix to compensate for this delay would be to set our system to measure PH value every 1 minute.

Conclusion:

The expected results were reached as specific pumps were turned on when ph readings were outside our desired range. PH UP pump turned on when PH probe was dipped in lemon juice (PH 2 < PH 5.5) and PH DOWN pump was turned on when PH probe was dipped in baking soda solution (PH 8 > PH 7). Although there was a delay in PH readings stabilizing when during sudden transition of the PH probe to another solution with differing PH level, it was only a max 1 minute delay and can be compensated by reading PH every 1 minute. One concern for this setup would be the noise of relay and wear of the relay if switched very often.

PH Sensing and Pumps With Solid State Relays

Goal: (CAUTION** HOUSEHOLD OUTLET LIVE AC VOLTAGE USED**)**

After initial testing with mechanical relays, a few concerns arise. Mechanical relay produces a substantial clicking noise when it switches and can be quite bothersome when switching frequently. Another concern would be the longevity of a mechanical relay as this kind of switch relay is mechanical, thus, prone to wearing out and shorter life span. For this part of the report, solid state relays will be tested instead of mechanical relay as a solution to these concerns.

SS Relay:

Pros:

Silent switching

No mechanical parts, no mechanical wear

Lower power consumption

Longer lifespan

Cons:

No noise for indication of switching

Best for AC Loads switching (faulty when loaded with motor speed controller directly)

Materials:

Raspberry PI 3b+

Waveshare High Precision AD/DA Board

Grove PH sensor

Solid State Relay Module

Jumper Cables and Alligator Clips

Peristaltic Pump

1N4007 Diode

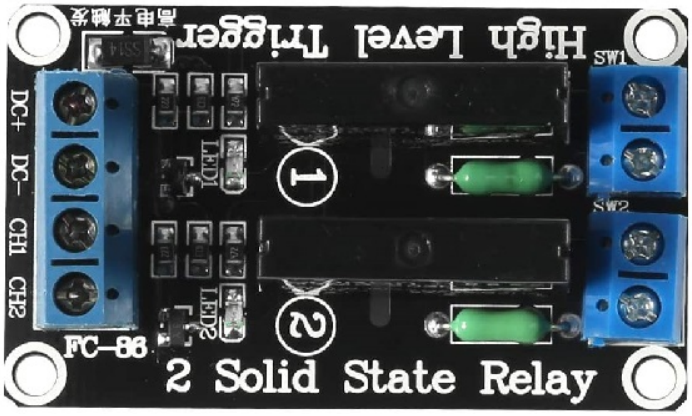
Motor Speed Controller

Screw Terminal Blocks

Outlet Duplex Receptacle

Optional: Junction Boxs for housing

Schematic: (CAUTION**BE CAREFUL HIGH LIVE AC VOLTAGE!!!!**)**

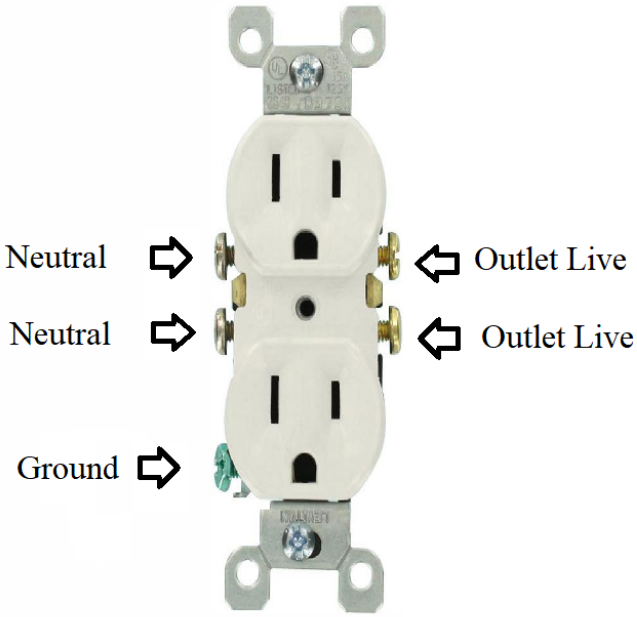


120V LIVE!

Outlet Live Connection

120V LIVE!

Outlet Live Connection



Procedure:

1. Connect the circuit as shown in the schematic diagram above. Junction box housing and screw terminal blocks are optional but recommended for extra safety and ease of connection.
2. Configure the code from section “PH Sensing and Pumps With Mechanical Relays” to accommodate the high trigger for the SS Relay module. Mechanical relays before are low triggers (on when GPIO signal is LOW). Replace all “GPIO.LOW” with “GPIO.HIGH” and “GPIO.HIGH” with “GPIO.LOW.”

```
def GPIOSetup():
    GPIO.setup(PH_UP, GPIO.OUT, initial = GPIO.LOW)
    GPIO.setup(PH_DOWN, GPIO.OUT, initial = GPIO.LOW)
if __name__ == '__main__':
    try:
        GPIO.setmode(GPIO.BCM)
        GPIOSetup()
        ph_sensor = phsensor.PHSensor(0,14) #phsensor object
        while True:
            #ph sensing from ad da board
            ph = round(ph_sensor.read_ph(0),1)

            #if ph level is below range
            if ph < PH_MIN:
                #turn on PH up pump
                GPIO.output(PH_UP, GPIO.HIGH)
                print(f"PH UP pump on at PH:{ph}")
                #time.sleep(pump duration)
                time.sleep(1)
                #turn off PH up pump
                GPIO.output(PH_UP, GPIO.LOW)
                print("PH UP pump off")

            #if ph level is above range
            if ph > PH_MAX:
                #turn on PH down pump
                GPIO.output(PH_DOWN, GPIO.HIGH)
                print(f"PH DOWN pump on at PH:{ph}")
                #time.sleep(pump duration)
                time.sleep(1)
                #turn off PH down pump
                GPIO.output(PH_DOWN, GPIO.LOW)
                print("PH DOWN pump off")
```

3. Run code as shown in “PH Sensing and Pumps With Mechanical Relays” but with modification from step 2.

Results:

As expected, the solid state relays switch the correct devices (pumps) when PH sensor readings are out of the desired range but produce no noise during switching.

Hardware

- **DAOKI 5A Motor Speed Controller Module**

Working Voltage:5V-35V (Peristaltic pump 12V)
Can be purchased here.

- **Grove PH Sensor (E-201C-Blue):**

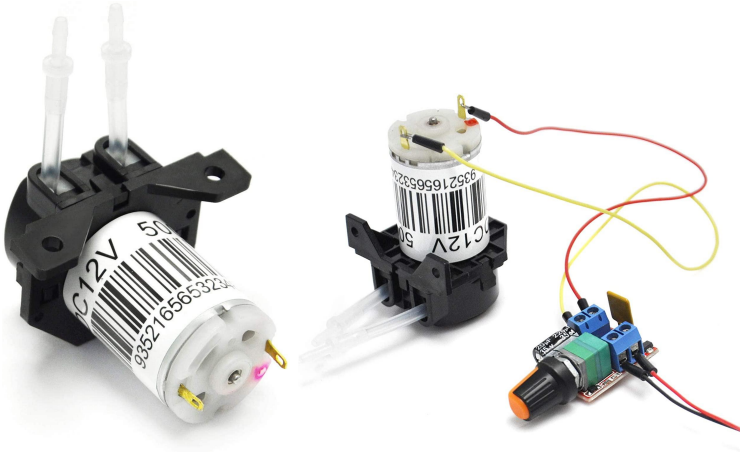
  Seeed's Grove PH module with a blue ph probe sensor. Since this ph sensor operates at 3.3V and 5V, Seeed claims that it works with Arduino and theoretically with Raspberry with additional ADC conversion with a Hat. No current demo code/instructions are provided at Seeed's site for Raspberry PI. More details at their site.

Features:

Operating voltage: 3.3V/5V

Resolution:±0.15PH（STP）

- **Peristaltic Pump**

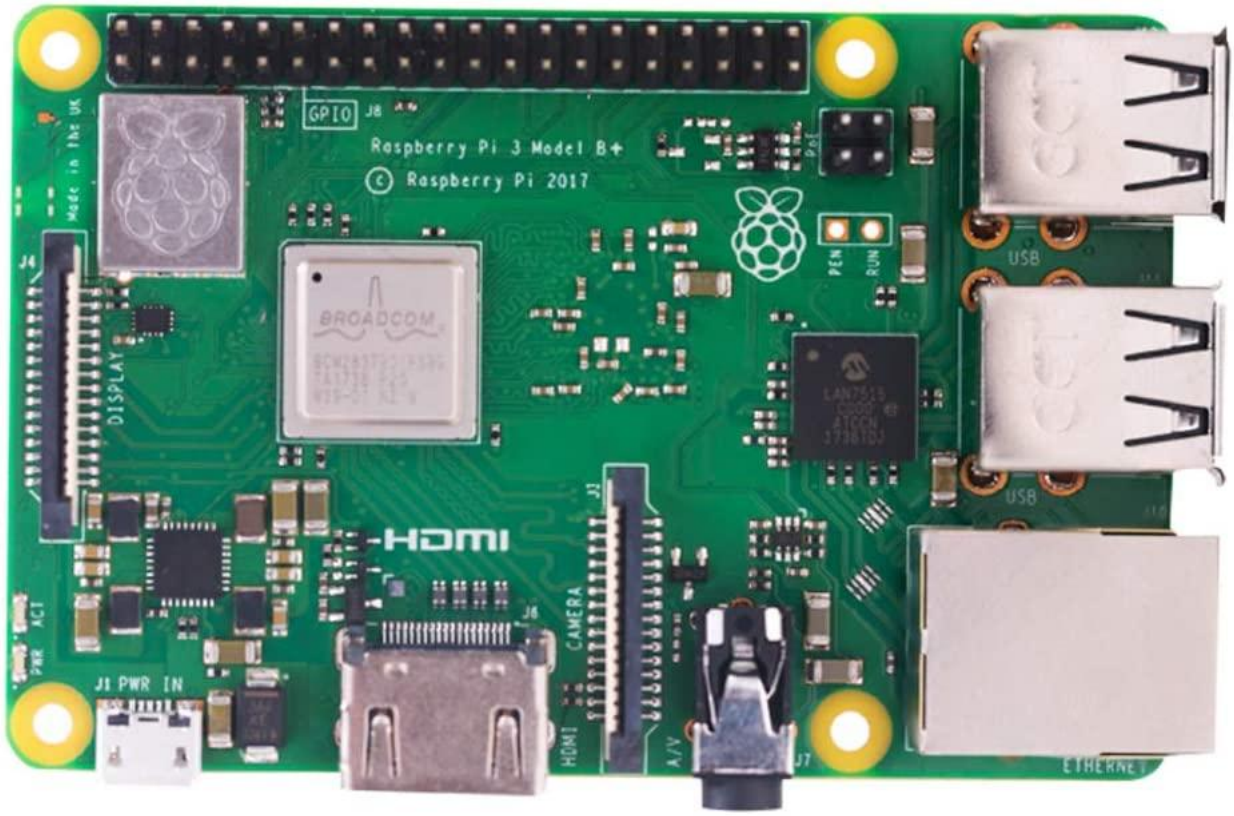


12V Peristaltic Pump

Can be purchased here.

❖ **Raspberry PI 3B+:**

Raspberry PI is a popular mini computer board great for developers. It sports GPIO pins to connect variety of circuit and electronic devices (ie. sensor, motors,..etc). Also, with a more powerful processor than Arduino, RPI can run more computation and work like a computer.



RP3B+ features:

CPU:Broadcom BCM2837B0 quad-core A53 (ARMv8) 64-bit @ 1.4GHz

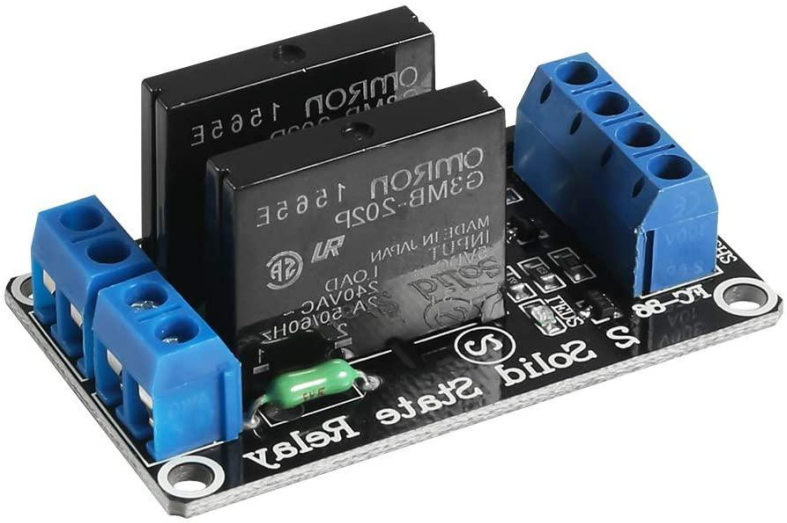
Network:Gigabit Ethernet (via USB channel), 2.4GHz and 5GHz 802.11b/g/n/ac Wi-Fi

Bluetooth: Bluetooth 4.2

Storage: Microsd card slot

Peripheral: USB, HDMI, Camera Serial Interface (CSI), Display Serial Interface (DSI)

❖ **2 Channel Solid State Relay Module 5V 2A High Level Trigger**



2A Solid State Relay Module

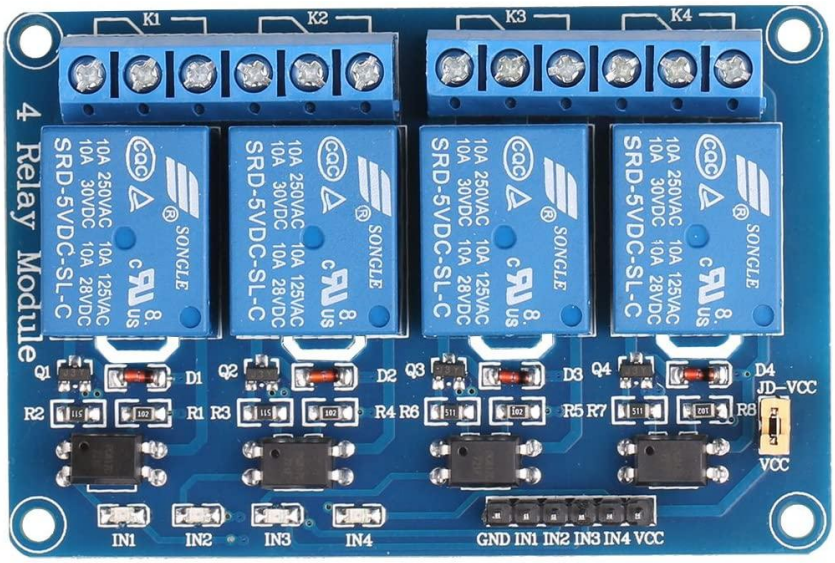
Input Power: 5V DC / 26.8mA (Turns on all channel)

High Trigger (Turns on GPIO.HIGH)

Best for Switching AC Load Up to 240AC

Can be purchased here.

❖ **4 Channel 5V Mechanical Relay Module**



High Current Relay, AC250V 10A ,DC30V 10A

Each Relay need 50-60mA Driver Current

Can be purchased here.

- **Waveshare High Precision AD/DA Board**

  Raspberry Hat board provided by Waveshare with the primary function of ADC/DAC conversion for the Raspberry PI board. It sports an impressive ADS1256 8 channel 24 bit precision ADC which is great for our application. Available here.

Features:

GPIO: 40 Pin GPIO

ADC:ADS1256, 8ch 24bit high-precision ADC (4ch differential input), 30ksps sampling rate

DAC:DAC8552, 2ch 16bit high-precision DAC

References

http://pss.uvm.edu/ppp/pubs/oh34.htm

http://www.electricalterminology.com/solid-state-relay-vs-mechanical-relay/