ME:4145 Industrial Internet of Things

Lab #6
Ultrasonic Range Sensor & Bluetooth

Introduction

The objective of this lab is use an ultrasonic sensor to measure distance and based off of that distance control a DC motor and LEDs. Bluetooth communication will be used to request data from the client. Like the WiFi communication lab, the RPi will be setup as a local server and your computer will be used as a client. In addition, threading in Python will also be introduced. This will allow tasks to be ran simultaneously in one program without being blocked by other functions or tasks. Example programs for the ultrasonic sensor, Bluetooth communication and threading are provide in the lab folder. Note, in the past for some operating systems, the Bluetooth module does not work properly. If this happens in your case, you can substitute the Bluetooth communication with WiFi (like the last lab).

Getting Started

What you will need

To complete this lab, you will need the following components:

- HC-SR04 ultrasonic distance sensor
- small DC motor (3-5V)
- resistors
- breadboard
- jumper wires
- tape measure or ruler
- computer with Bluetooth
- 1 × GREEN LED
- 1 × RED LED
- 1 × BLUE LED
- 1 × ORANGE LED (or substitute)

Ultrasonic Distance Sensor

An ultrasonic distance sensor is designed to sense object proximity or range using ultrasound reflection, similar to radar. The basic operation consists of a transmitter emitting a

high-frequency ultrasonic sound and listening for the sound to return. The distance to the object can be calculated using the definition of speed

$$Speed = \frac{Distance}{Time}$$

where $Speed$ is the speed of sound a, which is dependent on altitude, and $Time$ is the amount of time it takes for the sound to hit an object and return back to the receiver. Because the receiver senses the time to hit an object and return, we need to divide this time in half. Thus, the distance to the object is calculated as

$$Distance = \frac{a \times Time}{2}$$

Raspberry Pi connections

HC-SR04

The HC-SR04 Ultrasonic sensor for the Raspberry Pi has four pins: ground (GND), Echo Pulse Output (ECHO), Trigger Pulse Input (TRIG), and 5V Supply (Vcc) as shown in Fig.1. Connections from the RPi are 5V to Vcc, ground to GND, and an output GPIO

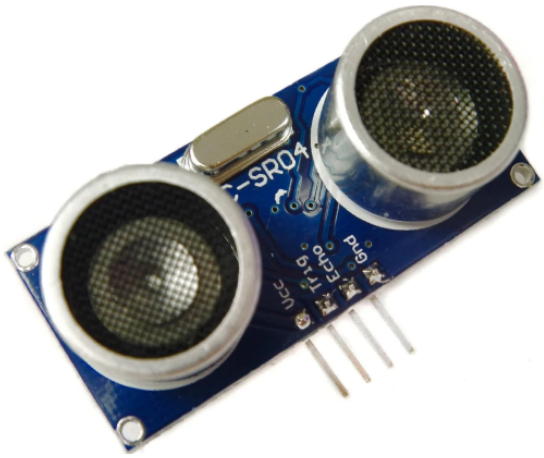

Figure 1: HC-SR04 ultrasonic sensor.

connected to TRIG to trigger the sensor and an input GPIO connected to ECHO to listen. To operate the sensor, the TIRG pin is set to HIGH for a sort period of time then set to low. The sensor then sets the ECHO pin to HIGH (5V) for the duration of the pulse. From here pulse waves bounce off any nearby objects with some being reflected back to the sensor. When the ECHO pin records the return of the sound wave it sets the pin back to LOW (0V). The time between the ECHO pin being HIGH and LOW is what we are interested in, i.e., $Time$.

IMPORTANT. As discussed above, the HIGH logic on the HC-SR04 is 5V. However, the HIGH logic on the RPi is 3.3V. Sending a 5V signal into that unprotected 3.3V input port could damage the RPi. To make this work we will need to use a voltage divider circuit, consisting of two resistors. This circuit will lower the 5V sensor output to 3.3V for the RPi.

The circuit for one type of voltage divider is shown in Fig. 2. With V_{in} and V_{out} known, you will need to determine the value of $R1$ and $R2$.

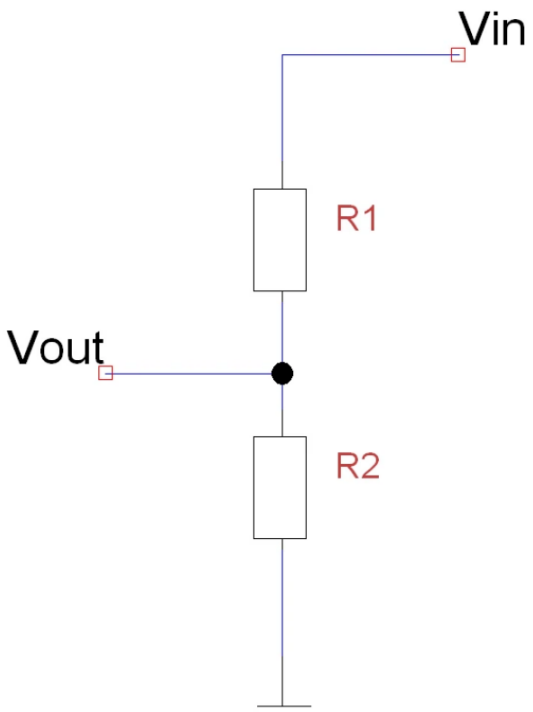

Figure 2: Voltage divider.

L293D and DC motor

Connections to the L293D and DC motor will be the same as those used in the previous laboratory. As a reminder, the wiring diagram is provide in Fig. 3.

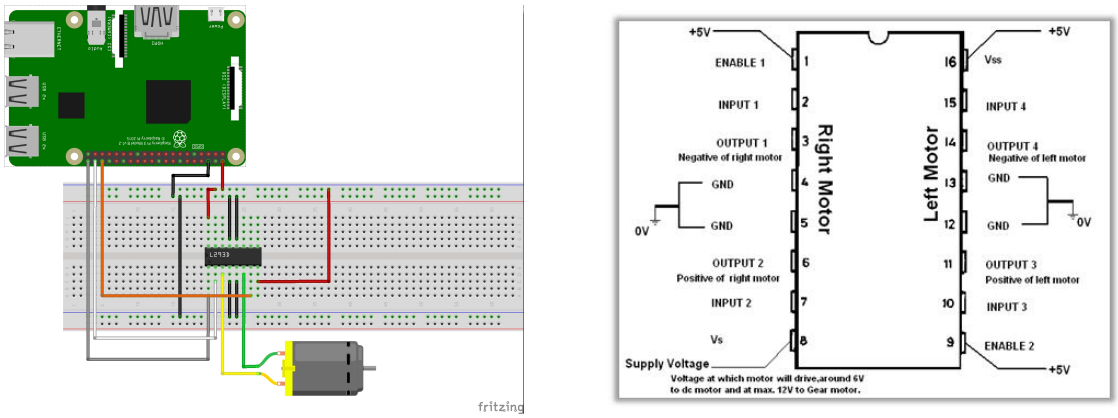

Figure 3: (a) L293D and DC motor wiring example (b) L293D diagram.

Task

Write a client and server Python scripts to perform the following:

1. measure the distance using the HC-SR04 of a movable object
    - use a tape measure or ruler to check for accuracy
    - move the object closer and farther away and repeat
    - take note of any errors or inconsistencies you encounter
2. set default proximity (distance) levels in your program: $L1$, $L2$, $L3$, and $L4$
3. have the following occur based off of the distance d, measured from an object:
    - $d > L4$: have the motor spin at 100% and flash a GREEN LED at 50% duty cycle. Here, a cycle is defined as 1 second, meaning the LED will be ON 0.5 second and OFF 0.5 second.
    - $L3 < d \leq L4$: have the motor spin at 75% and flash a BLUE LED at 50% duty cycle where a cycle is 1 second.
    - $L2 < d \leq L3$: have the motor spin at 50% and flash an ORANGE LED at 50% duty cycle where a cycle is 1 second.
    - $L1 < d \leq L2$: have the motor spin at 25% and flash a RED LED at 50% duty cycle where a cycle is 1 second.
    - $d \leq L1$: turn the motor OFF and turn a RED LED ON (continuously).
    - NOTE: your program should notice when the object is moved and act accordingly in real-time
4. Allow the client to modify the $L1$, $L2$, $L3$, and $L4$ limits
5. Have the program return to default limits at the clients request
6. Every 30 seconds, the client should request the distance and motor duty cycle
7. The client then needs to push this information to a Google Sheets Database recording the following (don't forget data headers):
    - Date: year-mm-dd
    - Time: hr:min:sec
    - Distance: you choose units
    - Motor speed: duty cycle
    - Current proximity limits: $L1$, $L2$, $L3$, and $L4$

Submission Materials

Upload the following to ICON no later than the due date:

1. Python script/s
2. A document providing the values of the resistors used for the voltage divider and a short justification as to why those values were chosen
3. Copy of your google sheet containing 15 minutes worth of data
4. A short video of you moving the object, displaying the distance and showing motor speed and LEDs updating
5. Bring your assembled circuit to the next lab and be prepared to perform tasks based off of requests from the instructor