Lab 3: Sense & Respond with Intelligent Control Structure

MECH2110
Semester 1 2019

Nicholas O'Dell, Johannes Hendriks, Craig Wheeler.

1 Introduction

It is important for automated systems to sense and respond to their environments. This lab covers the basics of responding to sensor inputs, and demonstrates a useful control structure called the state machine.

You are required to complete the following tasks.

- Regulate the speed of a stepper motor depending on a measured distance.
- Implement a state machine so that a series of tasks can be completed.

The lab is worth 4% of your course grade and is graded from 0-4 marks.

2 Sense & Respond (2 marks)

Task: Control a stepper motor according to the following set of rules:

- While the distance between the ultrasonic sensor and an object is greater than 200 mm, run the stepper motor at full speed, which for these steppers is 200 steps per second.
- While the distance between the ultrasonic sensor and an object is between 100 and 200 mm, operate the motor at a the following speed $200 \times \frac{d-100}{100}$, where d is in millimetres.
- When the distance between the obstacle and the sensor is 100 mm or less, stop the motor.

We will be using the *AccelStepper* library, which provides more functionality than the default Arduino stepper library that we used last week. The default stepper functions are 'blocking functions', meaning that the Arduino cannot command the steppers to run continuously while completing another task. The *AccelStepper* library contains non blocking functions, making it more suitable to this lab and possibly your Warman solution (Library documentation: `http://www.airspayce.com/mikem/arduino/AccelStepper/classAccelStepper.html`).

Create a new sketch and complete the following:

1) Download the AccelStepper library through the library manager within the Arduino IDE. Select **Sketch** → **Include Library** → **Manage Libraries**, and search for **accelstepper**, locate the **AccelStepper** library and select install.

2) Connect the stepper motor and ultrasonic distance sensor as shown below:

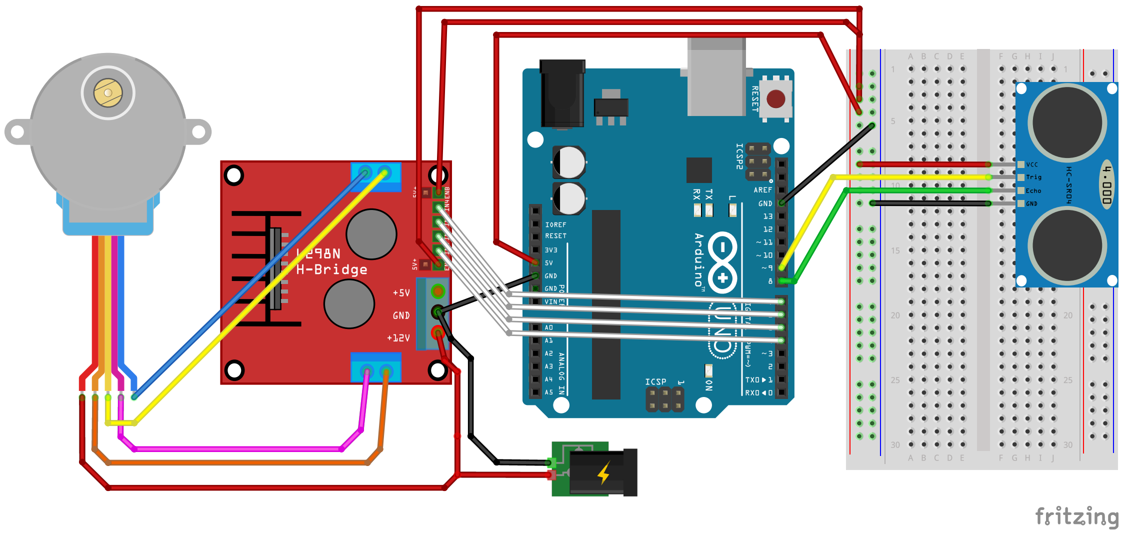

Figure 1: How to connect the stepper motor to the motor driver and Arduino: Connect the stepper motor's pink and orange wires to **OUT1** and **OUT2**, connect the stepper motor's blue and yellow wires to **OUT3** and **OUT4**, connect the stepper motor's red wire to power using the attached jumper wire. Ensure the **GND** terminal of the motor driver is connected to one of the Arduino's **GND** pins using the attached jumper wires. Connect **ENA** and **ENB** to **5 V** of the Arduino (as there are only two 5 V pins available you can either break out the power onto the breadboard or use the 5 V pins directly behind the ENA and ENB pins on the motor driver). Connect the pins **IN1-IN4** of the motor driver to pins **4-7** of the Arduino. Now connect the ultrasonic sensor: **VCC** to **5 V**, **GND** to **GND**, **ECHO** to pin **8** and **TRIG** to pin **9**.

3) Using an `if else` chain create a program that follows the rules outlined above. Start with the template code below (available in `lab3_files.zip` on Blackboard):

```
#include <AccelStepper.h>

#define ECHO 8
#define TRIG 9
#define SPEED_OF_SOUND 340.29 // m/s
//Declare stepper motor here using: AccelStepper
//e.g. AccelStepper myStepper(4,4,5,6,7); declares a stepper motor names
    //...myStepper with a 4 wire interface connected to pins 4-7
AccelStepper myStepper(4,4,5,6,7);

void setup() {
  // put your setup code here, to run once:
  pinMode(ECHO,INPUT);
  pinMode(TRIG,OUTPUT);
  Serial.begin(9600);
}
//variables for duration,distance,speed
long duration=0;
float mm=0;
```

```
int stepsPerSec=0;
void loop() {
  // put your main code here, to run repeatedly:
  // Read in a distance from the ultrasonic distance sensor:
  // The PING))) is triggered by a HIGH pulse of 10 microseconds.
   digitalWrite(TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG, LOW);
  //read length of time pulse
  duration = pulseIn(ECHO, HIGH);
  // convert the time into a distance
  mm = 0.5 * duration * 1e-6 * SPEED_OF_SOUND *1e3;
  Serial.print(mm);
  /*fill the if else chain:
  - Runs the motors at full speed if the distance is greater than 200mm
  - Runs the motors at a speed proportional to the distance read when between 100
      //... and 200 mm away
  - Stop the motors when less than 100mm away
  */
  if(/*insert test condition for distance > 200*/)
  {
    //set motors full speed(200)
    Serial.println(" Contstant Speed");
  }else if(/*test condition for distance >100*/)
  {
  stepsPerSec = 200*(mm-100)/100;
  myStepper.setSpeed(stepsPerSec);
  Serial.println(" Slowing Down");
  }else if(/*test condition for <100*/)
  {
    //set motor speed to 0
    Serial.println(" Stopped");
  }
  myStepper.runSpeed();
}
```

4) You will require these functions.

- `AccelStepper`: Initialise a stepper motor.
  Example: `AccelStepper myStepper(4,4,5,6,7);//initialises a stepper with a 4 wire interface`

- `setSpeed`: Sets motor speed.
  Example: `myStepper.setSpeed(200);//sets speed to 200 steps per second, use 200 as your`
  `//... max speed in this task.`

- `runSpeed`: This function must be called as often as possible and checks if it is time to step the motor or not.
  Example: `myStepper.runSpeed();`

You may like to refer to the following Arduino reference pages:

- https://www.Arduino.cc/en/Reference/If
- https://www.Arduino.cc/en/Reference/Else

Shown below is an example of how to initialise and run your motor at a constant speed:

```
#include <AccelStepper.h>

AccelStepper stepper; // Defaults to 4 pins on 2, 3, 4, 5

void setup()
{
   stepper.setSpeed(50);
}

void loop()
{
   stepper.runSpeed(); //This function must be called as often as possible
}
```

5) Demonstrate your program to your tutor.

3 Control Structure - State Machine (2 marks)

Task: Implement a state machine that accomplishes the same goals as the previous task. Then extend the state machine to wait for a time period and control a servo.

A state machine has the following components:

- A variable that stores the current state
- A switch-case within a loop to control the flow between states
- The states, which each have:
  - An action
  - One or more test conditions to decide if the state should transition.

An example of a simple state machine that controls an LED is shown below:

```
#define  LED_ON 0
#define  LED_OFF 1

void setup(){
pinMode(13,OUTPUT);
}

int state = LED_OFF;

void loop(){
  switch(state){
  case LED_OFF: //These instructions are executed if the state is set to LED_OFF
        digitalWrite(13,HIGH); //turn on LED
        if(true){ //if we want the state to change (always in this example)
        state = LED_ON; //set the state to LED on
        }
        break;
  case LED_ON:  //These instructions will be executed if the state is set to LED_ON
        digitalWrite(13,LOW); //turn off led
        if(true) //if we want the state to change (always in this example)
        {
        state = LED_OFF; //set state to LED_OFF
        }
        break;
  }
  delay(1000);
}
```

This simple example obviously does not require a state machine for only two states, but a state machine is easily expanded to include more operations. A state machine is perfect for the Warman competition because it allows you to split up the different tasks into clearly defined states. As an example, your states could be along the lines of: FOLLOW_EDGE_OF_TRACK, ROTATE, TRAVERSE TRACK, LIFT_LOAD, WAIT_FOREVER; where each of the transition/exit conditions between each of these states is determined by sensor data or time.

You may like to refer to the following Arduino documentation on `switch/case` statements: https://www.arduino.cc/en/Reference/SwitchCase.

1) Begin with the following example code where the first state has already been completed for you:

```
#include <AccelStepper.h>

#define CONSTANT_SPEED 0
#define SLOWING_DOWN 1
#define STOPPED 2
#define ECHO 8
#define TRIG 9
#define SPEED_OF_SOUND 340.29 // m/s

int state = CONSTANT_SPEED;

//Declare stepper motor here using: AccelStepper
//e.g. AccelStepper myStepper(4,4,5,6,7); declares a stepper motor names
     //...myStepper with a 4 wire interface connected to pins 4-7
AccelStepper myStepper(4,4,5,6,7);

void setup() {
  // put your setup code here, to run once:
  pinMode(ECHO,INPUT);
  pinMode(TRIG,OUTPUT);
  Serial.begin(9600);
}
//variables for duration,distance,speed
long duration=0;
float distance=0;
int stepsPerSec=0;

void loop() {
  // put your main code here, to run repeatedly:
```

```
  // Read in a distance from the ultrasonic distance sensor:
  // The PING))) is triggered by a HIGH pulse of 10 microseconds.
  digitalWrite(TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG, LOW);
  //read length of time pulse
  duration = pulseIn(ECHO, HIGH);
  // convert the time into a distance
  distance = 0.5 * duration * 1e-6 * SPEED_OF_SOUND *1e3;
  Serial.print(distance);

  switch(state){
  case CONSTANT_SPEED:
    Serial.println(" Constant Speed");
    myStepper.setSpeed(200);//Task for this state
    //TODO: insert test condition here
    break;
  case SLOWING_DOWN:
    Serial.println(" Slowing Down");
    //TODO: set speed here

    if(distance>200){//test condition to transition back to previous state
      state = CONSTANT_SPEED;
    }else if(distance<100){//test condition to transition to next state
      state = STOPPED;
    }
    break;
  case STOPPED:
    Serial.println(" Stopped");
    //TODO: set speed to 0
    //TODO: insert test condition here.
    break;
  }
  myStepper.runSpeed();

}
```

2) Demonstrate your working state machine to your tutor **(1 mark)**.

3) Add 2 more states that:

   i) Hold the motor for 5 seconds after stopping.

   ii) Then rotates a servo motor that was initially at 0 degrees to 90 degrees.

4) Demonstrate your extended state machine to your tutor **(1 mark)**.

4 Recommendations

- **Important:** Next week's lab, Lab 4, has mandatory pre-lab. If you arrive at your lab without completed pre-lab you will not be permitted to complete the lab.
- Think of how you can separate the Warman course into different tasks/states, and how you can complete each one.
- Brainstorm with your team various mechanical designs, and think about how they might work with various sensors and actuators.