Lab 5

ECEN 2270
University of Colorado Boulder

April 29, 2022

Luke Hanley
Nicholas Haratsaris
Ginn Sato

Introduction

Our final lab was definitely the most fun, as well as the most rewarding. Before we began working on the lab, we decided that our main goal was to implement an ultrasonic sensor to avoid hitting obstacles, and a piezo speaker working in conjunction with the ultrasonic sensor to employ proximity beeping. First we had to implement these devices in hardware, which was relatively simple, and then we had to implement them in software, which was somewhat more difficult. Given the nature of the Arduino IDE software, we were able to find a lot of example code online to assist with our software development. Along the way, we realized that the speaker wasn't loud enough, and, like any engineer has to do, we had to scrap the speaker idea and transition to something new- in our case, automatic headlights and turn signals.

Equipment List

- OSEPP Electronics LTD HC-SR04 Ultrasonic Sensor Module
- Sparkfun Photoresistor
- Texas Instruments TLC3702 Operational Amplifier
- Diodes Incorporated ZVN2106A N-Channel MOSFET
- 4 pin push button
- Sparkfun Resistor Kit
- Sparkfun Capacitor Kit
- Digilent Analog Discovery 2
- Fully Assembled ROB 0025 Robot
- Breadboard Wires and Jumper Wires
- LTSpice Software
- Arduino Nano Every with USB connector
- Arduino IDE Software
- Microsemi Corp. 1N5818 diode
- Sparkfun LED Kit

Part 1: Implementation

Our first step in implementing the HC-SR04 ultrasonic sensor was to look at the datasheet and understand how it works. Basically, you trigger the sensor, and then it echoes a voltage that stays high for a time proportional to the distance in front of it. We implemented this into our software by writing a check distance function. We thought of two ways to implement distance response into the robot's movement. One method was to have a continuous turn- that is two states: object within distance threshold, turn, else, go straight; the method being, object within distance threshold, turn left 90 degrees, then go back to straight. The former created a smoother turn, where the robot would turn just enough to avoid the object, sort of like a roomba. The second created a more constructed robot movement, in which we were able to implement more randomness into the movement, by randomly choosing to turn left or right 90 degrees.

We then implemented the piezo speaker, which was relatively simple. We were even able to find source code online to play specific tunes- such as the Darth Vader theme, or the Super Mario Bros. theme song. Given the state machine we created, it was relatively easy to have the speaker beep whenever the distance was below the threshold. However, we simply could not hear the speaker over the whirring of the motors. We even built a voltage controlled current

source circuit with an op amp and MOSFET to draw higher amounts of current straight from the battery through the speaker, in which the Arduino was the voltage controller. However, this was to no avail, and eventually we made the decision to scrap the speaker idea and move onto something new. We had a limited amount of time, and we didn't think the speaker was "cool enough" to warrant sinking all of our time into. We found a photoresistor in one of our kits, and went from there.

We decided to use the photoresistor to implement automatic headlights on the robot, like newer cars have. The photoresistor acts like a short when it's light out, and an open when dark. Thus, we built a voltage divider from 5V on the arduino with a series photoresistor and 4.7k resistor. The resistor was chosen to be able to handle the full 5V across it, which would only result in 5mW power in the resistor. The voltage node in between the two components was read into one of our analog pins. Given the 10 bit ATMega processor, we would read a value from 0 - 1024 at this pin. A high value corresponds with brightness, because the photoresistor acts like an open, and a low value corresponds with darkness, because the photoresistor acts like a short. This is also known as a pull-down resistor configuration. We set our threshold to be roughly 300, which meant that if the robot went under the circuits lab benches the threshold would trigger. We would then set one of our digital pins high, which would power two parallel LEDs on the front of the robot.

The final implementation we made was a turn signal of sorts. This was relatively easy, since we simply had to include an LED blink in our turning functions. Each of our turn functions had some sort of delay, somewhere in the range of milliseconds. To implement the blinking, we simply had to set the LED high/low, delay, set the LED opposite, and use the rest of the delay.

We had two main software configurations, each with their own merits. The first only ever turned left, and was more of a state machine. This robot was better at navigating obstacles and tight spaces. In turn, it only had a left blinker. The second robot would go straight and if distance was below threshold, it would randomly call either a turn left 90 degrees or turn right 90 degrees function. It pretty much only had one state, but if it saw an object it would change its course. This configuration presented some element of randomness, as well as blinkers on both sides.

Our robots worked very well. If we let them loose in the laboratory they were very unlikely to run into obstacles. This was what we cared most about, and thus we were very happy that our robot would avoid obstacles and navigate through spaces. While it was doing all of that, it was blinking to indicate turns, and if it was dark enough it would turn on its headlights. Somewhat funny is the fact that even if the room was pitch black the robot would still navigate through, since its distance sensing was done by sending sonic waves- visible light has nothing to do with it.

Part 2: Future Implementations

If we had more time, we probably would have built some sort of mechanical body or shell, just to add another layer of protection to the robot. Given that our sensor was in the front

middle of the robot, sometimes the front corners would get caught on a wall, and then we would have to run in and turn off the robot. This challenge could be overcome by adding another sensor, so that we would have an ultrasonic sensor on the front left and front right of the robot, to really increase its sensing capabilities. This would also require a lot of software changes, but would help decide whether the robot should turn left or turn right.

We also wanted to utilize interrupts, but we couldn't find a good way to do it, since the sensor had to be triggered manually. We were going to do it with the photoresistor, but in the end it was just easier to implement as a polling function, not to mention we only had analog pins left.

Part 3: Recap

The biggest thing we learned was that sometimes you have to abandon an idea because it is unfeasible, and it's a better use of your time to work on something else. We learned this with the piezo speaker. Additionally, the speaker was probably the most difficult part of the lab, partly because of the noise, but also because it was difficult to implement into our movement functions/loops.

The most successful part of the lab was the ultrasonic sensor. It worked exactly how it was supposed to, and we were able to find example code online, not to mention that the Arduino IDE software has a lot of built in functions which made the implementation very easy. The photoresistor was also very successful, and we were able to implement automatic headlights relatively easily.

If we had a chance to do things differently, we would have not spent any time on the speaker, and probably would have tried to implement bluetooth control in addition to the ultrasonic sensor. If we had done this from the get go we probably could have achieved both goals.

Conclusion

Our advice to future students of 2270 is to start the labs early and pay attention from the start. Each lab builds on the previous one therefore applying yourself from the get go will give the best results. This is a 3-credit hour course that feels like 5 at times, however if you put quality work in, you will get quality knowledge and experiences out. This course is an opportunity to apply the many foundational skills learned in Circuits 1 and 2. It is an extremely rewarding course to take within the electrical engineering curriculum.

At its foundation, this course taught us the process behind constructing large projects with integrated circuits. We began the class classifying the motors of our robot, then we developed an in depth understanding of the components we would use, before integrating them into larger purposed circuits. At each point in the development of our robots we used circuit analysis techniques to choose various component values, and confirmed our results using simulation software.

This course allowed us to apply fundamental electrical engineering principles, and we will carry the ideals and concepts learned into the next courses, and hopefully into the industry. We are

grateful to have been able to take this class in an in-person environment, even though that meant countless hours and long nights spent in the laboratory. It was those long nights and frustrating hours of debugging that really made us feel like engineers. This course invoked our excitement for real, hands-on electrical engineering, and we can't wait for what the future has in store.

Appendix

```
while(1) {
  check_distance();
  check_light();
  if(distance < 17) {
    digitalWrite(FWD_Right, HIGH);
    digitalWrite(BACK_Right, LOW);
    digitalWrite(FWD_Left, LOW);
    digitalWrite(BACK_Left, HIGH);
    digitalWrite(blinker, HIGH);
    delay(125);
    digitalWrite(blinker, LOW);
    delay(125);
  }
  else {
    digitalWrite(FWD_Right, HIGH);
    digitalWrite(FWD_Left, HIGH);
    digitalWrite(BACK_Left, LOW);
    digitalWrite(BACK_Right, LOW);
    digitalWrite(blinker, LOW);
  }
```

Continuous Turn Software Implementation

```
  while(1) {

  rando = random(100);                    // randomly choose either 0 or 1
    rando = rando % 2;
    check_distance();                     // poll distance
    if(distance < 15) {                   // if reach treshold,
      if(rando == 1){
        TurnRight(90);                    // turn right
      }
      else{
        TurnLeft(90);                     // turn left
      }
    }
    else {
      digitalWrite(FWD_Left, HIGH);       // else continue straight
      digitalWrite(FWD_Right, HIGH);
    }
    if(analogRead(PhotoRes) < 500){       // check if it is too dark
      digitalWrite(LedLeft, HIGH);        // turn on headlights
      digitalWrite(LedRight, HIGH);
    }
    else{                                 // else it is bright
      digitalWrite(LedLeft, LOW);         // keep headlights off
      digitalWrite(LedRight, LOW);
    }

  }

}
void encoder_irq() {
  enc_pulse_count++;
}
```

Randomly Turn Right or Left Software Implementation

```
void check_distance() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);

  distance = duration / 148.0; //gives distance in inches
}
```

Check Distance Function

```
void TurnLeft(int degree){
  // Turn clockwise
   enc_pulse_count = 0;
    do {
    digitalWrite(FWD_Right, HIGH);
    digitalWrite(BACK_Right, LOW);
    digitalWrite(FWD_Left, LOW); //Adjust direction
    digitalWrite(BACK_Left, HIGH);
    delay(200);
    digitalWrite(LedLeft, HIGH);
    delay(200);
    digitalWrite(LedLeft, LOW);
  } while(enc_pulse_count < degree*9 );

    digitalWrite(FWD_Right, LOW);
    digitalWrite(BACK_Right, LOW);
    digitalWrite(FWD_Left, LOW);
    digitalWrite(BACK_Left, LOW);
    delay(300);
}
```

Turn Left Function Implementation

```
void TurnRight(int degree){
  // Turn clockwise
   enc_pulse_count = 0;
    do {
    digitalWrite(FWD_Right, LOW);
    digitalWrite(BACK_Right, HIGH);
    digitalWrite(FWD_Left, HIGH); //Adjust direction
    digitalWrite(BACK_Left, LOW);
    delay(200);
    digitalWrite(LedRight, HIGH);
    delay(200);
    digitalWrite(LedRight, LOW);
  } while(enc_pulse_count < degree*9 );

    digitalWrite(FWD_Right, LOW);
    digitalWrite(BACK_Right, LOW);
    digitalWrite(FWD_Left, LOW); //Adjust direction
    digitalWrite(BACK_Left, LOW);
    delay(300);
}
```

Turn Right Function Implementation