Computer Engineering Project I

Prepared for

Jalili Boudjadar

Associate Professor, Electrical and Computer Engineering
Aarhus University

By

Andreas Kaag Thomsen, Asger Song Høøck Poulsen, Kasper Svenningsen

Undergraduates, Electrical and Computer Engineering
Aarhus University

"Insert date here.."

1 Week 1

1.1 Introduction

This week we were introduced to the course and the upcoming project we would be assigned to do.

1.2 Groups

We were told to make groups of three for next week. Since the other class was not here, we could not continue any further and class was dismissed...

2 Week 2

2.1 Introduction to Arduino

This week we were introduced to programming on an Arduino. We used the Arduino IDE to write the code which would be deployed on an Arduino which was fixed to circuits system on a Breadboard.

2.2 Sensors

Today we added an ultra-sonic sensor to the circuits system to measure distance to an obstacle. The goal was to make the built-in LED blink for 1/10 of a second when an obstacle was less than 30 cm away but more than 25 cm away, 3/10 of a second when an obstacle was less then 25 cm away and more than 20 cm away and if the distance between the obstacle and the sensor was less than 20 cm away the LED should turn on constantly.

2.3 Errors

We succeeded at getting readings, but these were not the exact distance between the sensor and an obstacle. We made use of the data we received from the sensor which got printed as an output on the Arduino IDE screen. We assumed that our voltage input was the same as the distance to implement the see-think-act cycle. Even though this was not exactly the correct way to do this, it gave us an idea on how the Arduino worked with the sensors.

2.4 Remember for next week

We have not successfully come to a conclusion with the scaling factor.

3 Week 3

3.1 Fault in Arduino

Our Arduino's yellow LED did not work, so we got another Arduino, but this Arduino would not connect to our computer, so we could not push our updated code to the new Arduino. So we went back to using the Arduino from before and implemented an external LED for sensor nr. 2.

3.2 Goal for today

The goal was to make multiple range sensors work so we could measure distance from more than one direction. We did so by connecting the external LED to sensor nr. 2 and connecting the built-in LED from the Arduino to sensor nr. 1.

3.3 Using multiple connections in a single system

To connect the two sensors we made a chain connection on the circuit system so the sensors would not intefere with eachother. We did this by looking at the datasheet which showed us the following:

We updated the code on the Arduino so that it said which direction it should go in. E.g. if sensor 1 = front sensor was getting close to an obstacle we would print "go backwards". This would lay the foundation for when we are getting wheels.

3.4 Remember for next week

We applied the scaling factor but the accuracy was not spot on.

The next method is AN Output Constantly Looping. The first sensor will range, then trigger the next sensor to range and so on for all the sensor in the array. Once the last sensor has ranged, it will trigger the first sensor in the array to range again and will continue this loop indefinitely. Below is a diagram on how to set this up.

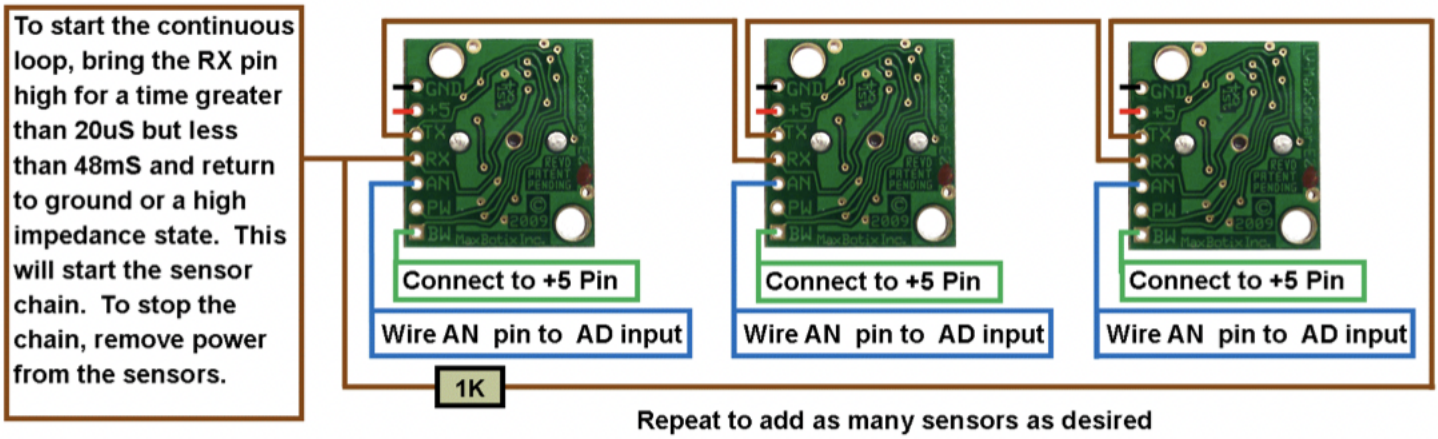

Figure 1: chain connection implemented on circuit system

4 Week 4

4.1 RGB Sensor

The mission of this week was to integrate an RGB sensor in the setup built in the previous weeks (Arduino-Breadboard-RangeFinder). The purpose of RGB sensors is to read dark tags from the floor, simulating a victim, during navigation. If a reading is not conclusive, a movement simulation should be calculated to step back/forth for a new reading to confirm/deny the presence of the dark tag.

4.2 Thoughts

We set a boarder of random boarder of 30 lux which would represent a target. If the sensor read between 30 - 40 lux the program was set to represent this as inconclusive and should simulate a step back/forth for a new reading to confirm/deny target. This was just to get a base of the coding.

4.3 Setting boarders for when target is found

The average value of illuminance in the room was calculated to 165 lx. These calculations were done while taking 36000 measurements and calculating the average of these.

The average value of illuminance with a dark tag 3 cm over the sensor was calculated to 22 lx. The largest value was measured at 30 lx and lowest value was measured at 12 lx with a standard deviation of 2.64 lx. These calculations were done while taking 49000 measurements and calculating the average of these.

So we set the value of our average + standard deviation = 24.64 lx as our boarder of "this is definitely a target". And between this value and our max value + standard deviation = 32.64 lx would be our questionable measurement.

4.4 For next week

We began installing virtual machines on Andreas' and Kasper's computers for next week.

5 Week 5

5.1 Ubuntu and setting up for RPI

We made some finishing touches for setting up the virtual machine with Ubuntu.

5.2 Python

We got further introduced to Python and its syntax.

6 Week 6

6.1 Connecting ubuntu to the internet

We had troubles last week connecting ubuntu to the internet which got fixed by removing our static IP. After, we finished last week's assignment which

was to write a small python code that creates an empty file in a folder of our choosing.

6.2 Learning ROS

We got introduced to ROS (Robot Operating System), and got thrown into the tutorial where we installed and configured our ROS environment. Thereafter, we did some small exercises to learn more about using ROS. We got stuck in exercise 2, so we have to reinstall ROS and do it all over again next time.

7 Week 7

7.1 Learning ROS

ROS got reinstalled and we began the tutorial again. This time it worked and we got through the tutorial. We were instructed to complete all exercises until 17.

7.2 Preparation to Intro to Robot programming

We began reading a Git repo on turtlebot navigation but did not understand much. We were told that we would get proper introduced to this next week. So hopefully we will understand the code in the Git better next time.

8 Week 9

8.1 First navigation example

Through today's slides we setup the workspace and began a basic navigation example that made the turtlebot go forwards until it met an obstacle at a min_distance then it would stop.

8.2 Edit navigation example

After getting the first example to work, our job was to edit and play with the code so that it would turn left, right or back up depending on which

angle we would meet our obstacle. Just like we did in week 3 with the print statements.
We ended up just playing with the code and getting some better understanding and ideas on how the code for the final project should work, but we did not end up with any final results.

8.3 For next week

1. We are not quite certain if our if-else statements are 100% correct.

2. We can not get the turtlebot to go backwards every time we want it to.

3. We are not quite certain that the indexes of our distances matched their corresponding angles.
   A solution to this could be to make our array be a construction of tuples? So it would look a bit like:

   array[(angle(x),distance(x)),(angle(y),distance(y)), ...]