Control of Mobile Robotics CDA4621
Fall 2017
Lab 2 Kinematics

Lab Report

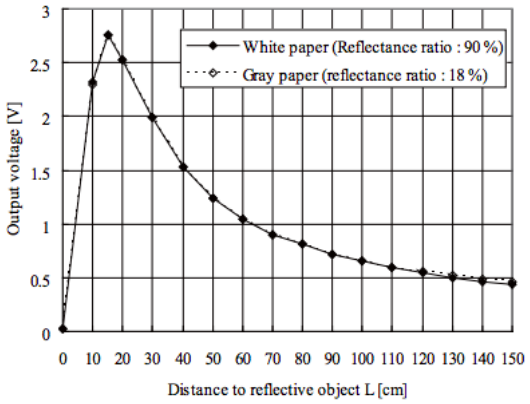
Code Files Included:

All files need the “MySharpSensors.h” file and “MyEncoders.h” file to compile.

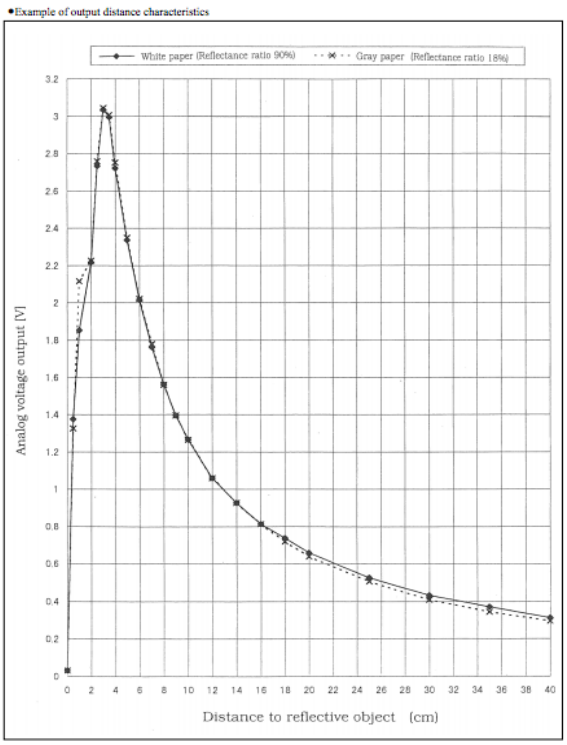
WallDistance.ino	Implements Task 3. Uses PID controller to keep robot 5 inch away from a wall at all times
WallFollowing.ino	Implements Task 4. Uses PID controller to follow an object on it’s right.

Converting Raw values to Distance:

Before we could get started with this lab, we needed to calculate the distance values from the sensors. There were many ways to do this but two stood out. Noting down raw sensor values at known distances and interpolating or using the graph of voltage output against distance to surface provided in the data sheet to make an equation which will give us the distance. We decided to use the data sheet graph for our calculations as we believed it will be more accurate.

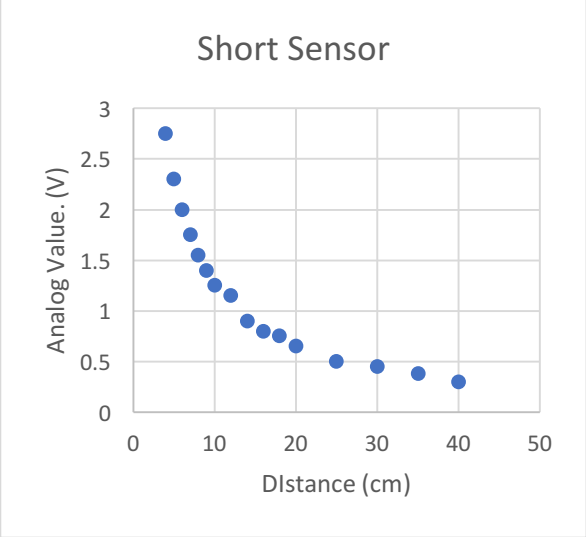


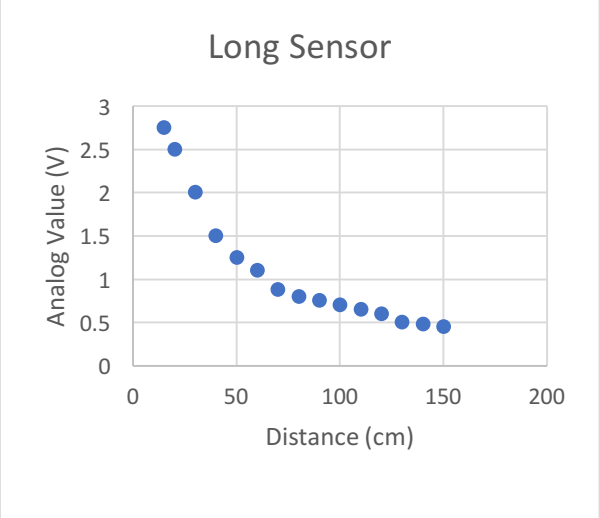
Datasheet Graph for Long Range Sensor



Datasheet graph for Short Range Sensor

Our next step was to convert the relevant part of these graphs into excel plotted graphs and find the equations of the lines. These are the two graphs we got.





We further converted our voltage from analog to digital using the resolution of Arduino's analogRead function. We use the following equation to convert the values.

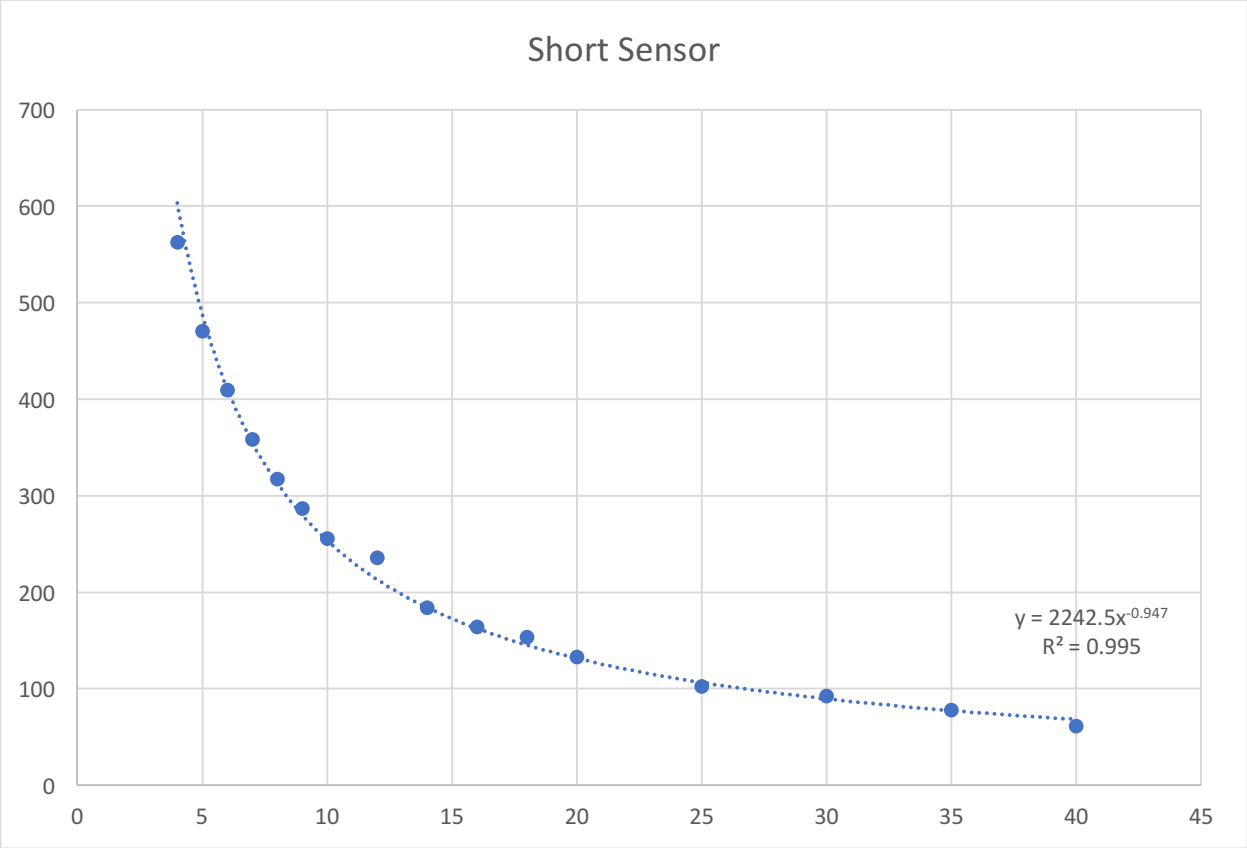
$$\frac{Resolution\ of\ the\ ADC}{System\ Voltage} = \frac{ADC\ Reading}{Analog\ Voltage\ Measured}$$

Since we have a 10 bit ADC with a 5 V power supply, we can simplify to:

$$\frac{1023}{5} = \frac{ADC\ Reading}{Analog\ Voltage\ Measured}$$

We now get our final graph with the equation of the line:

Short Range Sensor

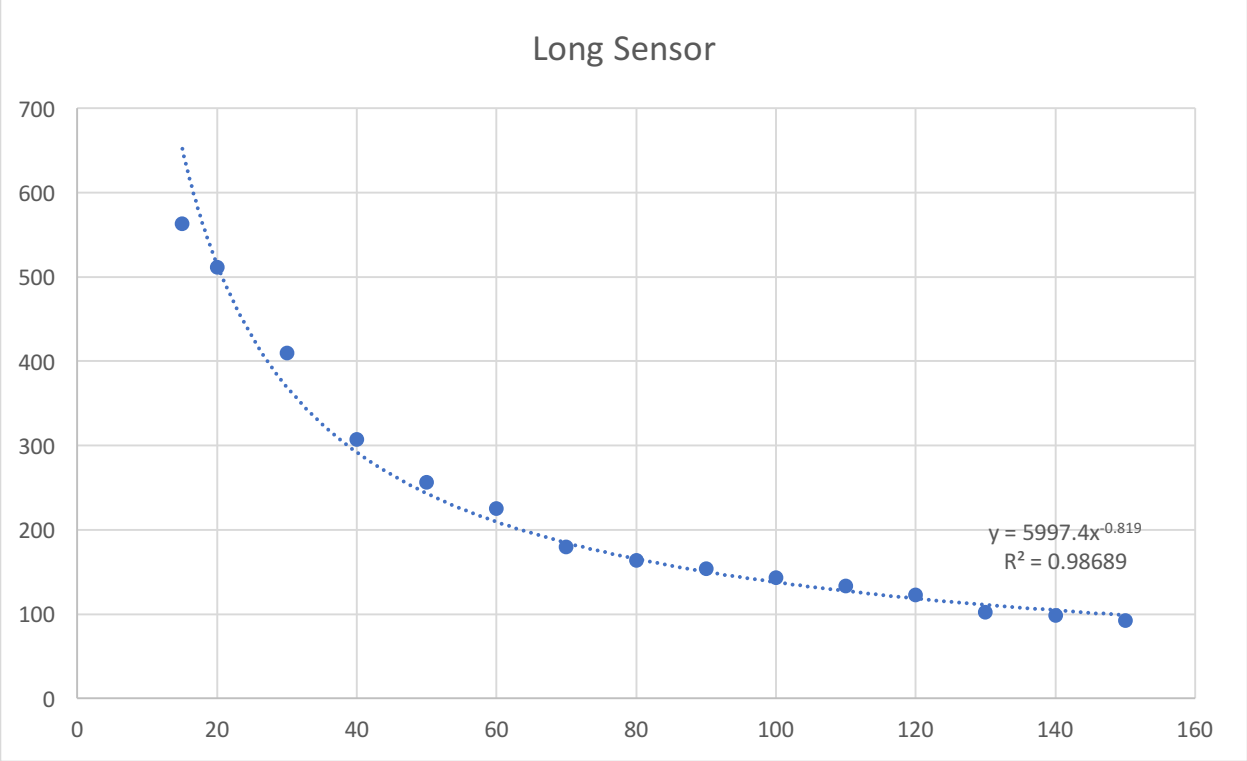


We then solved our equation for X to get:

$$x \approx \frac{3453.51}{y^{1000/947}}$$

This equation will give us the distance for any analog voltage value where X will be in cm and y in Voltage. To convert this to inches, we can divide the equation by 2.54.

Long Range Sensor



We then solved our equation for X to get:

$$x \approx \frac{41\,010.7}{y^{1000/819}}$$

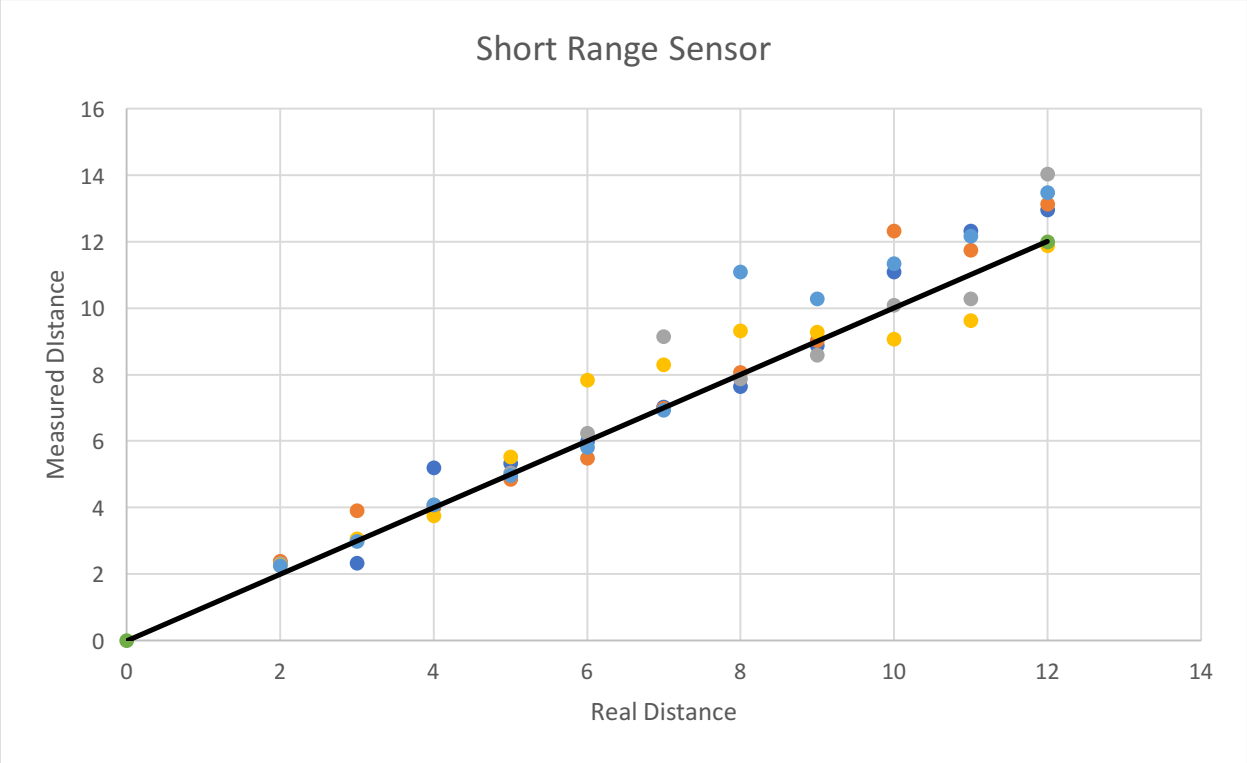
This equation will give us the distance for any analog voltage value where X will be in cm and y in Voltage. To convert this to inches, we can divide the equation by 2.54.

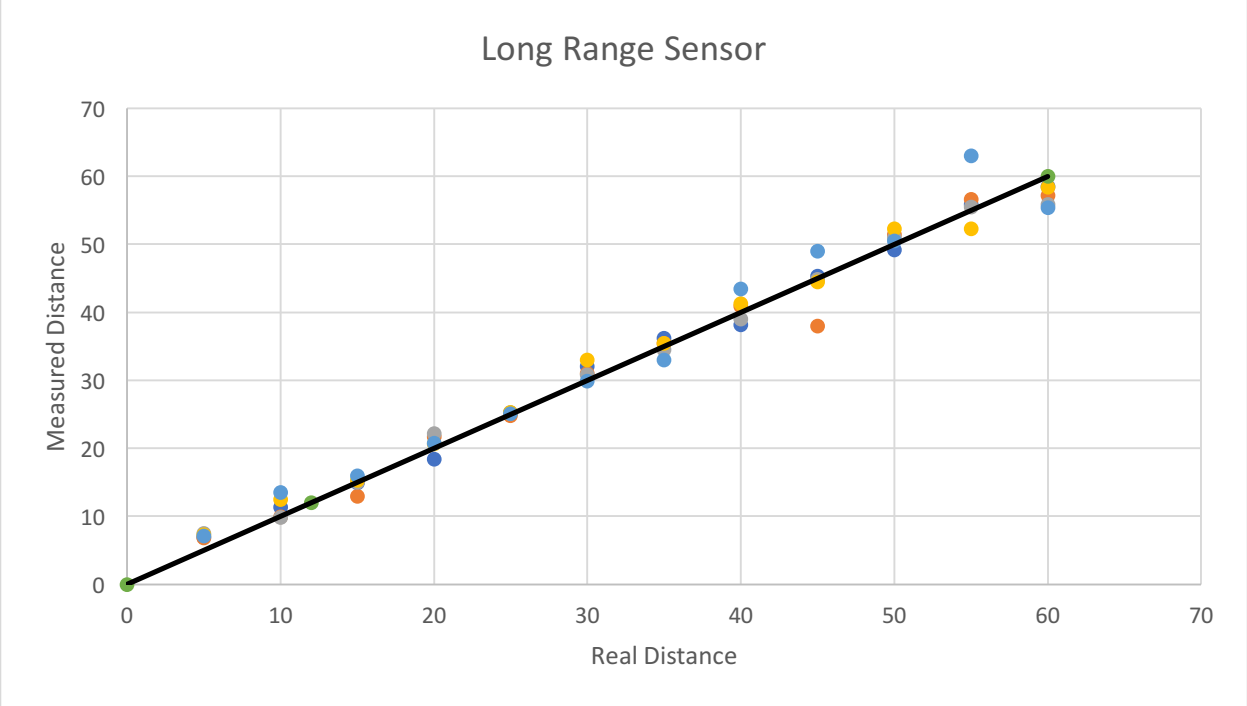
Now, we are ready to implement shortToInches and longToInches:

```
float shortToInches(int value) {
  float inches;
  inches =  1359.649606 * pow(value, -1.055966 ) ;
  return inches;
}
```

```
float longToInches(int value) {
  float inches;
  inches= 16145.945 * pow(value, -1.221);
  return inches;
}
```

Task 1:
We measured the distance from 2” to 12” five times for the short sensors and measured the distance from 5” to 60” five times for the long range sensor. Here are our results:

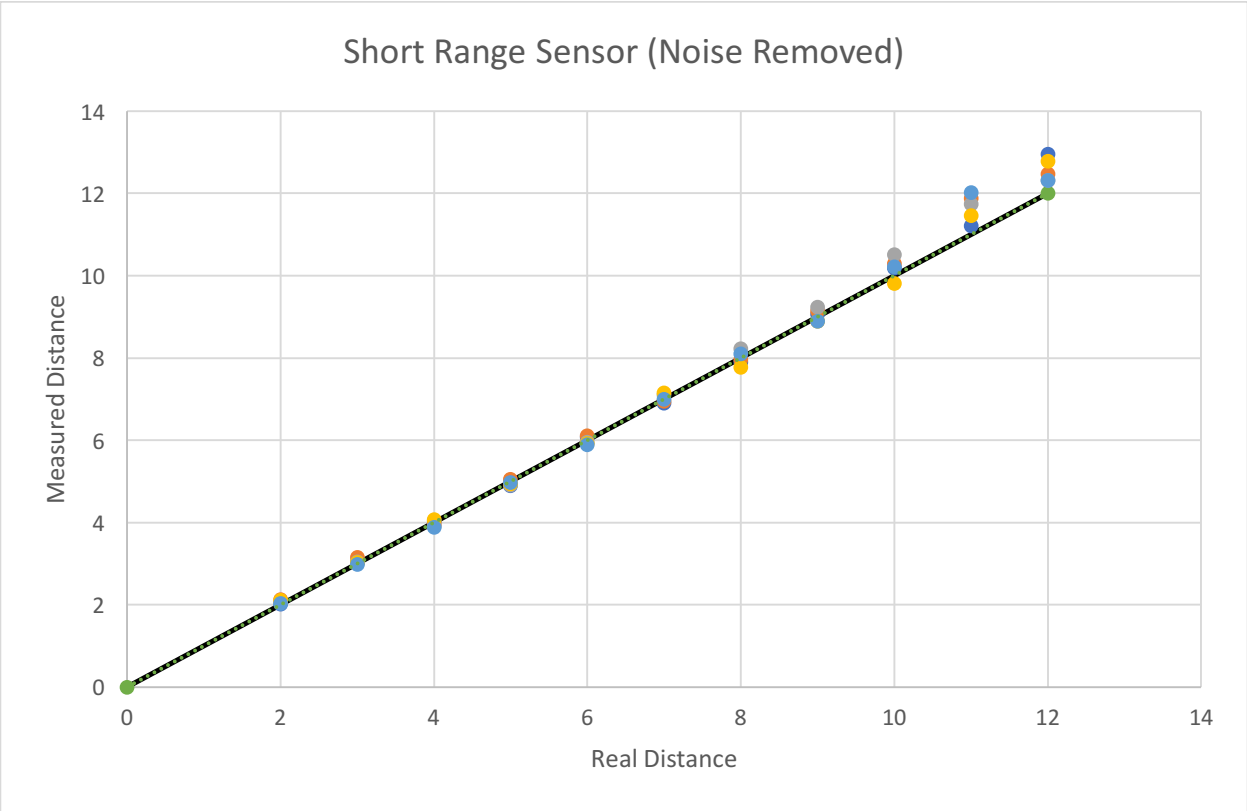


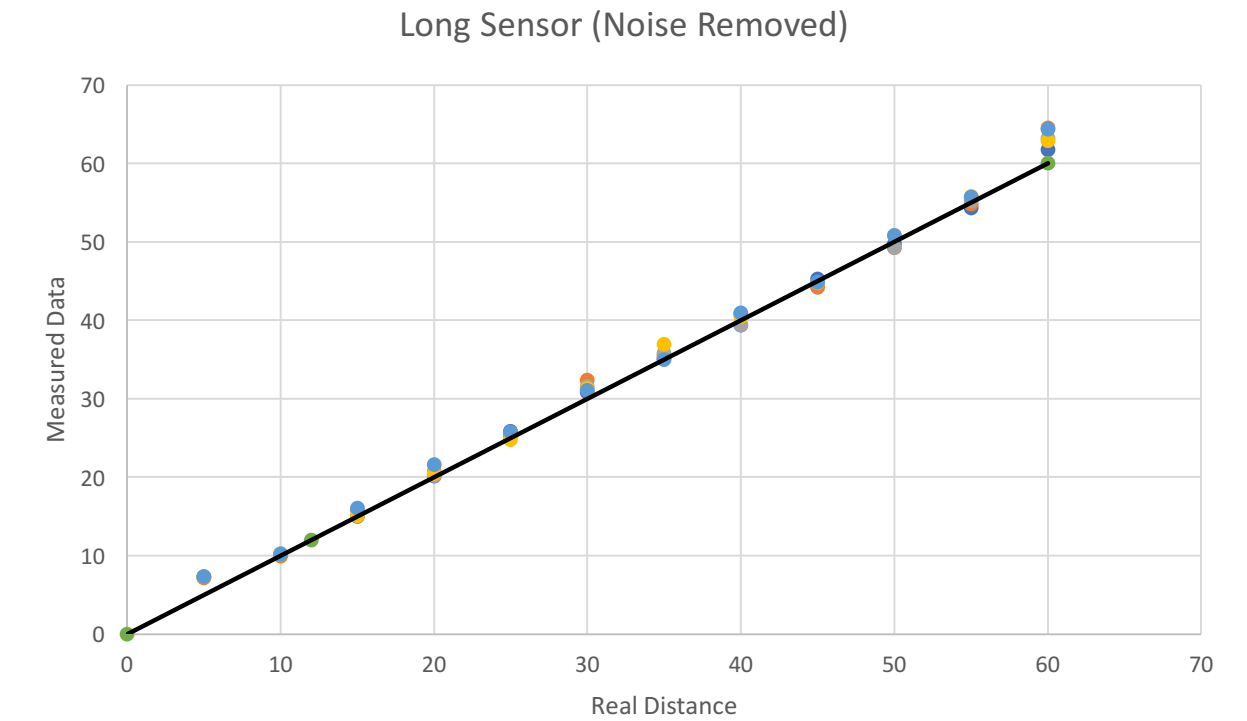


Not all measurements fall on y=x and Big intervals between values.

Task 2:

We then used medians and plotted 5 medians against known distances again





Now, the intervals are shorter and values are closer.

Task 3 -

$$u(t) = K_p * e(t) \quad \text{(Eq. 1)}$$
$$u_r(t) = f_{sat}(u(t)) \quad \text{(Eq. 2)}$$
$$u_r(t) = f_{sat}(K_p * e(t)) \quad \text{(Eq. 3)}$$
$$e(t) = r(t) - y(t) \quad \text{(Eq. 4)}$$
$$u_r(t) = f_{sat}(K_p\,(r(t) - y(t))) \quad \text{(Eq. 5)}$$

where:

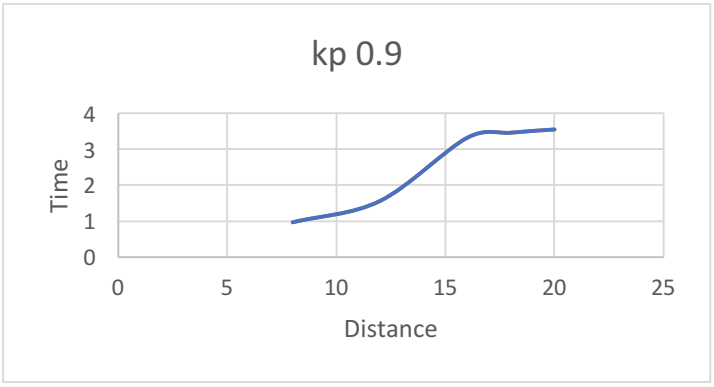
$r(t) = desired\ distance\ to\ the\ goal$
$y(t) = distance\ from\ robot\ to\ the\ goal$
$e(t) = distance\ error$
$K_p = proportional\ gain\ or\ correction\ error\ gain$
$u(t) = control\ signal\ corresponding\ to\ robot\ velocity$
$f_{sat} = Saturation\ Function\ (See\ explanation\ below)$
$u_r(t) = control\ signal\ corresponding\ to\ saturated\ robot\ velocity$
$u_r(t)$ → setSpeedsIPS(u_r, u_r) //function implemented in the first assignment

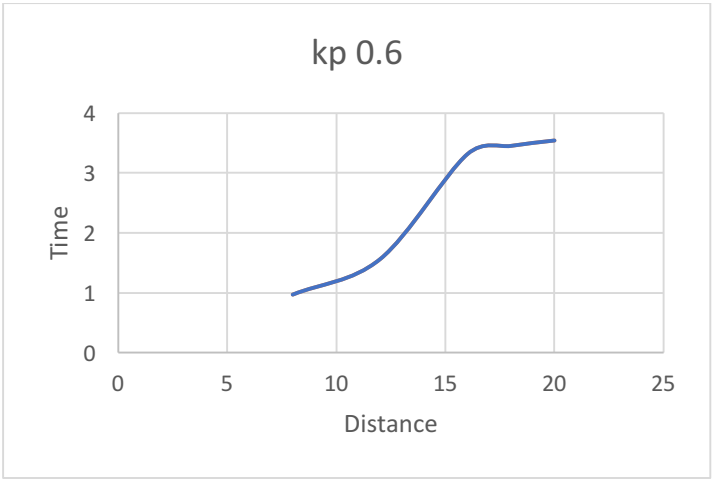
On the formulas above, f_{sat} is a function that limits the inputs to the function "setSpeedsIPS".

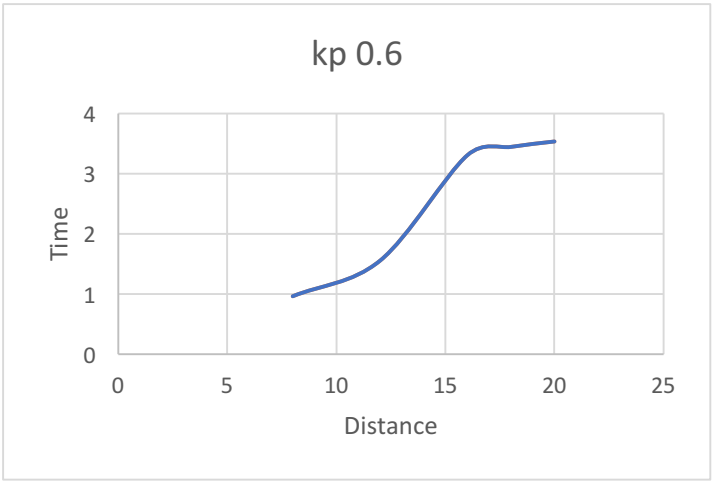
$$f_{sat}(x) = f(x) = \begin{cases} -6 & if \quad x < -6 \\ x & if \quad -6 \leq x \leq 6 \\ 6 & if \quad 6 < x \end{cases}$$

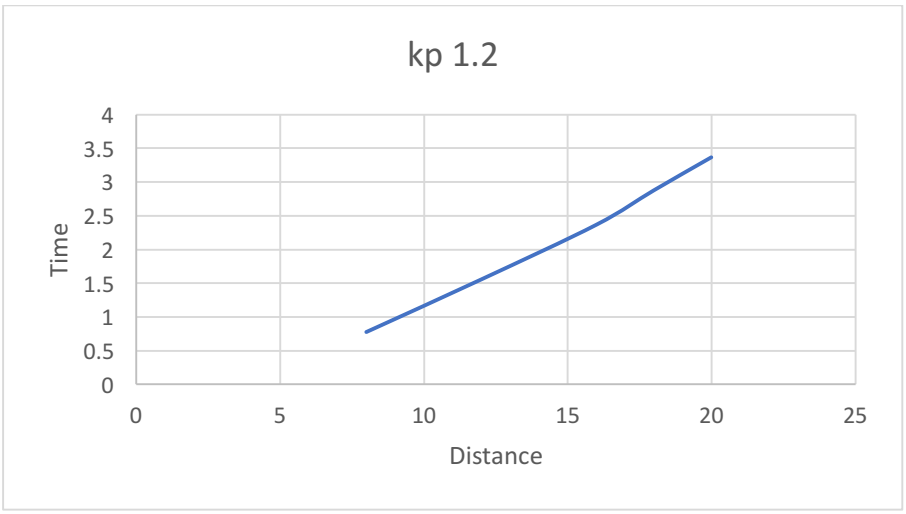
Equation 4: e(t) is the error. That is, it is the difference between the desired distance and the actual distance from the goal.
Equation 1: u(t) is used to control the velocity of the robot. It is the product of the error and the Kp. Kp determines the ratio of output to response to the error rate.
Equation 2, 3 and 5: ur(t) is the velocity that we want our robot to attain.

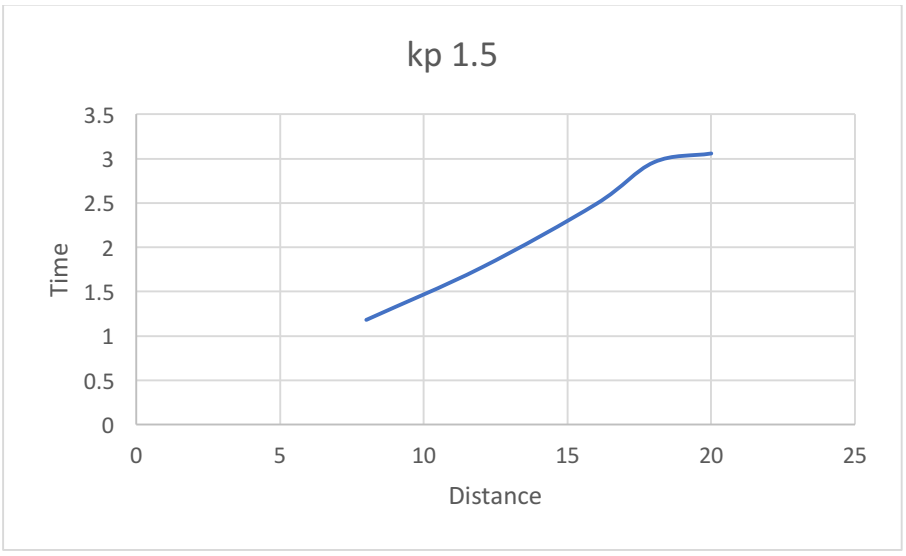
Task 4:


kp 0.9
Time
4
3
2
1
0
0
5
10
15
20
25
Distance


kp 0.6
Time
4
3
2
1
0
0
5
10
15
20
25
Distance


kp 0.6
Time
4
3
2
1
0
0
5
10
15
20
25
Distance


kp 1.2
Time
Distance
0
0.5
1
1.5
2
2.5
3
3.5
4
0
5
10
15
20
25


kp 1.5
Time
Distance
0
0.5
1
1.5
2
2.5
3
3.5
0
5
10
15
20
25


kp 5
Time
Distance
0
0.5
1
1.5
2
2.5
3
3.5
0
5
10
15
20
25

Conclusions:

Using the equations of the graphs was a good idea because our sensors seem to work well. Taking a median is even better as it decreases the margin of error and makes our answer more possible and plausible. This is evident from the graph as all the values are closer to each other in case of medians.

I would not use Kp below 1.2 at al. We had good, proportional results with Kp 5 and Kp 1.2. I will choose either one of them for my demonstration.