Ballistic Chronograph

Mech 307

December 10, 2021

Mechatronics - Dr. Fankell

Charlie Sanders
Jake Pollard
Kirk Russell

Table of Contents

Design Summary

As part of the course requirements for Mech-307 Mechatronics, students are tasked with the unique opportunity to develop a device without target functionality, but must include an output display, manual user input, automatic sensor, actuators/mechanisms, and hardware.Our group has taken this opportunity to design and develop our chronograph. This will utilize all of the required components and a few of the optional as well. We will be using an LCD, photoresistors, LEDs, a servo, potentiometer, 3D printed housing, and an active buzzer.

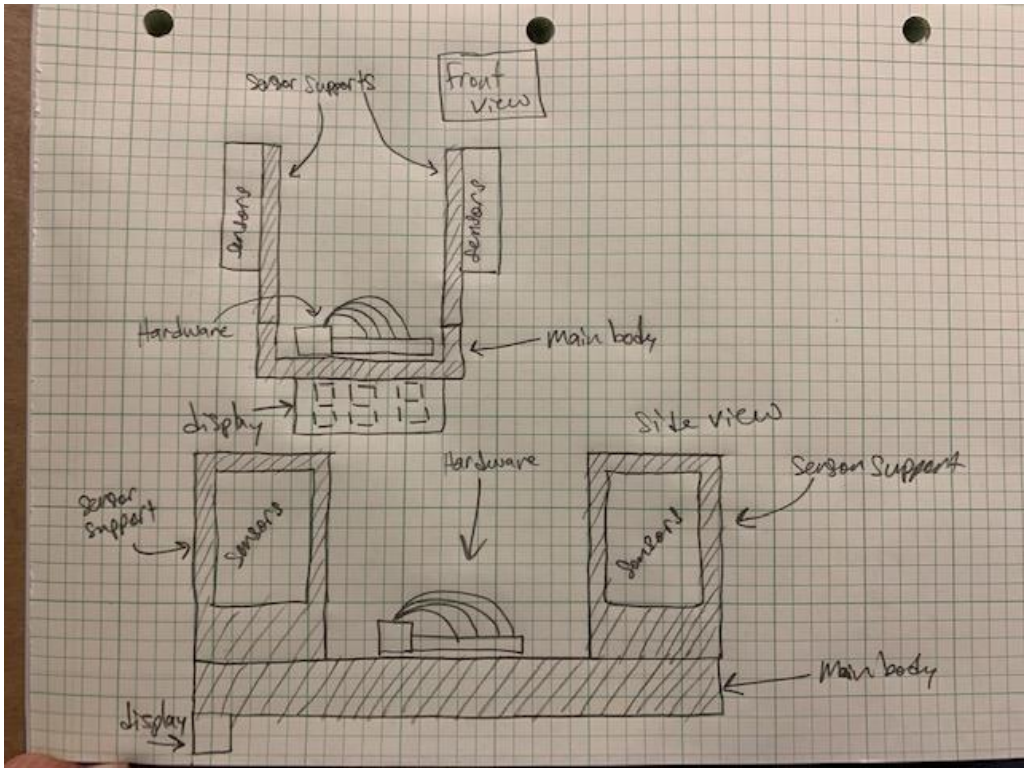

Figure 1: Example Chronograph

System Details

When initially tasked with this project there were many different variables we had to consider. One of the greatest was the cost of the device. While ideally, we would like to use the highest quality sensors and materials to develop a machine like this, we are students and do not have access to a larger budget that might be available to those in the industry. Other considerations were size, accuracy, utilization of required components, and difficulty. Listed below on a scale from 0-5, 5 being the most important, are the areas we believe we're the most integral to this project.

Cost: 3

Cost is an undeniable limiting factor for this project. While it is unacceptable to sacrifice functionality for the sake of cheaper materials, the cost of specific project elements has to be considered. As a general rule, it was decided that if an element is provided in the Arduino kit or available in the lab surplus supplies, it will be used in the project unless it reduces the ability of the chronograph to perform accurately and efficiently. The cost of the project is important, but the performance of the device is the top priority.

Size: 2

Due to the purpose of a ballistic chronograph, the size of the device as a whole is not extremely relevant to the functionality of the device. The chronograph size must be

logical. To measure velocity accurately, the device must have a large enough measurement window to collect data at 2 points of the projectile's path. If the 2 points are too close together, the velocity reading could be impacted by random errors caused by the environment. If the 2 points are too far apart, the velocity reading would act like an average velocity, and not provide insightful data on any specific point in the projectile arc. Besides the size of the measurement window, the structure of the chronograph will be what best accommodates the size-sensitive elements.

Accuracy: 5

The accuracy of the chronograph is the top priority of the project. The elements used to measure the velocity of the projectile will be some of the few elements that extra money will be spent on to ensure high-quality measurements. In addition to high-quality measurement devices, the construction of the chronograph will be focused on eliminating any sources of error that could potentially impact the validity of the data collected.

Utilization of required components: 5

Meeting the requirements of the project guidelines is of equal importance to the design process as is the accuracy of the device. The construction of the device will revolve around achieving accurate measurements using all required components of the project. After the basic function of the chronograph is achieved, additional parts will be incorporated that satisfy the more advanced aspects of the project guidelines.

Specifications

Engineering Specifications

1. Accuracy
2. Required components
3. Size of Device
4. Cost
5. Durability
6. Distance between Antenna
7. Speed of Deliverable
8. Device Completed on Schedule

Trade-Offs

Mounted Plugged in Chronograph

The obvious advantage of an outlet-powered chronograph is the constant power supply. When troubleshooting the device, the question of if there is power supplied would never be an issue. However, a limitation is that you are limited in where the chronograph may be used then. One must be able to plug the device into an outlet for the chronograph to turn on. This means that spontaneous in the field use is nonexistent.

Independent Battery Powered Chronograph

Conversely, the beauty of a battery-powered chronograph is that it can be used anywhere you go. This ability to not need an outlet is a major pro for this setup. However, it too has its downsides. While its portability is a greatly welcomed feature, it does bring up reliability concerns if the battery were to die. When constructing the device itself, there too could be the question of if the device is not in operation due to a power issue.

Phototransistors

Phototransistors and diodes, although similar, do have some key differences. A benefit of phototransistors is that they have greater light sensitivity. In addition, they also generate both current and voltage which can then be used to interpret the inputs

measured. The largest downside to phototransistors is that they have a slower output response which is important to keep in mind for our testing and prototyping.

Photoresistor

A benefit of photoresistors is that they have a much faster output response. This is very nice for our purpose of a chronograph. However, they have less light sensitivity, and only generate current as a response.

System Details

Sensors

The biggest area that this project leans on is the accuracy of the sensors that are measuring the propelled object through the two terminals. For this reason, the majority of research time is spent on sensors. Initially from our knowledge from class and quick research, we thought of using photoresistors and phototransistors to measure the speed of the traveling object. However, since the accuracy of the sensor readings was integral to this project, and after further research, we discovered that these types of sensors reached their peak limits relatively quickly and would not be able to accurately measure smaller objects traveling at high velocities. Take, for example, an object of 5mm in length traveling at 1,000 m/s will only be in front of the sensors for a few microseconds; which a standard phototransistor would not be able to accurately measure. For this reason, we landed on the decision to use photodiodes configured in a photoconductive, or reverse biased, setup. While this setup uses a higher voltage of 27 V, photodiodes can have a sub-microsecond response which was integral to our project. Diodes of this nature typically operate in the near-infrared range of 890 nm wavelength, much faster than standard visible light.

Controller

Now that we decided on a sensor that would be able to accurately measure our projectile, we needed a processing/control unit that could handle these kinds of inputs. The Arduino Uno provided to us in this class has a digital sampling rate of 2 Mhz and a maximum clocked resolution of ½ microsecond. These reading speeds are only achievable by using a whole register and parsing the results instead of the traditional digitalRead() function provided in the Arduino IDE library.

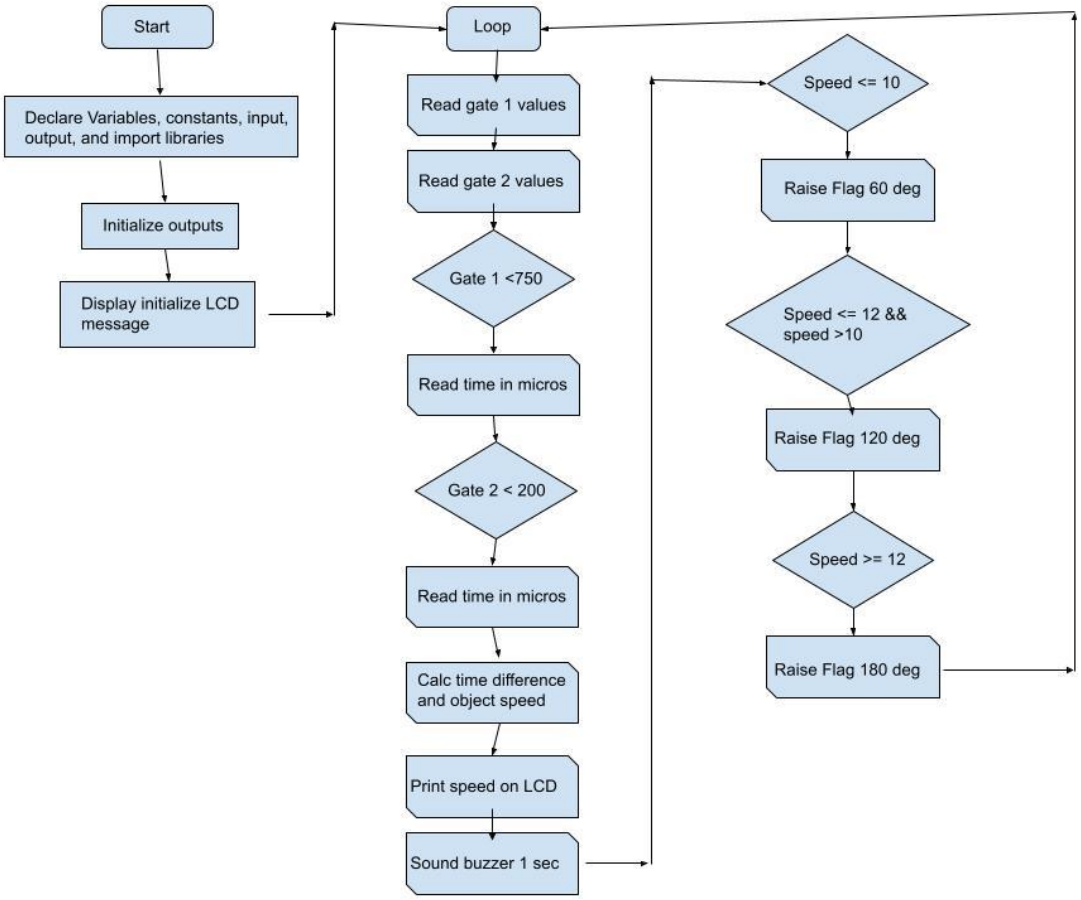

Figure 2: Code Flowchart

Casing

For the frame and casing of the device, I would like to go through 2 different versions depending on the time we have after development for a better-defined result. In the first iteration, I would like to use wood for everything. Not only do all of our group members have extra wood that we can use for this project, to help keep costs down, but it is also a material that is extremely easy to work with and we can manufacture the sizes and parts very quickly. After the development of a working prototype, I would like to design a more aluminum and 3D printed refined version. This will help with the overall aesthetic of the device as well as clean up some of the mistakes we may have made in our prototype.

Display

For our display, we will be using the single LCD keypad shield screen which comes included with the Arduino kit. This was an easy choice for several reasons. One we already have, which allows us to minimize costs. Secondly, it is the perfect size screen for this project. We do not need to output that much information to the user so a single screen output is all we need. This also allows us to save space and have overall lower power consumption.

Design Evaluation

A. Output Display

Our project uses a liquid crystal display for outputting the speed of the object traveling. This was used in one of our group homework assignments however we went a step a little further with having a constant message displayed on the top line to give a message to the user and the lower line which displays the speed of the device will stay until the next time an object passes through the gate which will cause it to overwrite what is currently displayed.

B. Audio Output Device

We decided to add a buzzer to the device for those who are using the device by themselves. If you are further away and have an object pass through we thought it was important to inform the user the device had functioned properly so in the off chance it did not function properly the user does not come to the device to find it did not produce a reading.

C. Manual User Input

For interaction with the device, we added a potentiometer that interacts directly with the LCD. This allows the user to control the contrast of the display so the user can adjust it to their liking depending on their surrounding environmental conditions for optimal reading.

D. Automatic sensor

This component is more or less the heart of our project. Without the use of photoresistors, we would not be able to measure the object moving. The overhead LEDs produce a constant light that the code knows an average

baseline reading of and once that light screen is tripped(the light intensity dips below a certain threshold) the rest of the code is tripped and rolls into action.

E. Actuator, Mechanisms & Hardware

For this section, we opted to design our case in SolidWorks and 3D print our case. This allowed us to rapidly prototype the design we wanted and eliminate the initial prototyping time to help reduce cost and time. This also allowed us to create much more accurate parts than hand-made or hand-machined components.

A SG90 servo motor was also added to give more feedback to the user, as a device like this is typically used from a bit of a distance. The servo was added with a flag that will be raised a certain amount depending on the speed of the object passing through the gates. If the object's speed is less than 10 m/s the flag raises 60 degrees, if the object's speed is between 10-13 m/s the flag is raised 120 degrees, and if the object is traveling faster than 13 m/s the flag is raised a full 180 degrees. This feature was added to help give more immediate feedback to the user so they did not have to get up and check the output speed every time to get a rough idea of the speed of the object from a distance.

F. Logic, Processing, and Controls

Our project uses an open-loop control setup. This allows our device to run completely by itself with much input from the user. After each run, the device is immediately ready to go again allowing the user to perform as many runs or tests as they would like without having to reset anything. AS well as calculating the speed of the object based on the time spent between two gates at a distance using a simple physics equation. The biggest miscellaneous component used in

this project was the use of unsigned variable types with our micros() which allowed us to store twice as much useful data as a standard long by eliminating the use of negative values.

Partial Parts List

Product: Chronograph	Date: 10/7/21

Item #	Qty	Name	Description	Cost per Unit	Total Cost
1	1	3D Printed Housing	PLA was chosen for its versatility, ease of printing, and availability	$20.49	$20.49
2	4	Photoresistor	These small photoresistors change resistances depending on the amount of light it is exposed to	$1.50	$6.00
3	2	Diffused LED - White 10mm	The large size and opaque color of these bulbs allows for a consistent more soft light	$0.55	$1.10
4	1	LCD1602 module with pin header- LCD Display	A multi-use programmable LCD display	$12.35	$12.35
5	1	10k Potentiometer	A multifunction device that allows user input to vary the resistance	$1.25	$1.25
6	1	Arduino UNO	The heart of the system that all software and hardware is run through	$22.95	$22.95
7	1	Active buzzer	The very small buzzer that can be implemented as an alarm or for many purposes in a circuit	$1.95	$1.95
8	1	Servo Motor SG90	Small but powerful servo with a duty cycle of 1-2 ms	$5.61	$5.61

Lessons Learned

The biggest struggle we faced in the design of this device is realizing how much refresh rates are. When measuring an object traveling 10 to 25+ m/s across such as a short distance of ½ a meter, having systems and code that can process the information fast enough is extremely important.

The first adjustment I had to make after I had created the initial test circuit involving just one photoresistor and an overhead LED was the refresh rate from the baud rate from Serial.begin() that we had been using in the lab of 9600. I saw there was a good amount of lag once we began looking at the sensitivity drop when objects began moving quicker and quicker and in some cases, while watching the serial monitor the graph I could see would visibly lag. My fix for this was increasing the data rate to 115200. This sampling rate increased our data received by a factor of 12 and allowed me to get a much more accurate reading from the photoresistor. By finding the average steady-state reading of the photoresistor when exposed to the LED I was able to trigger the first loop once it dipped below a certain value (the object breaks the light screen causing a shadow across the photoresistor dropping the reading being sent to the Arduino).

Next was the struggle of using the built-in function millis(). The first problem was I could not perform any type of logic using it when I stored it to an int variable type. I found out this was because millis() required a variable type with a much higher size and greater range. Below is a chart of the types of the main variables used in Arduino.

Type:	Size(byte):	Range:
boolean	1	0 to 1
char	1	-128 to 127
int	2	-32,768 to 32,767
long	4	-2,147,483,648 to 2,147483,647
float	4	3.4028235E-38 to 3.4028235E+38
double	4	3.4028235E-38 to 3.4028235E+38

Chart 1

After doing this I decided to go with a long variable type which ended up working well, but then I found an unsigned long. This variable type only counts positively from zero effectively doubling your positive value range because it won't store negative values. This was perfect for my application because we would never be dealing with a negative time value from millis().

One of the final coding changes we made was to use micros() instead of millis(). This allowed us to get even more accurate timing data from the gates. We went from our timing being 1 milliseconds resolution to a resolution of 4 microseconds using our 16 MHz Arduino. The only downside to this was the issue with overflow and data stored from this higher processing rate. Using our variable type of unsigned long we will reach overflow with micros() after 71.6 minutes. This means the device will need to be powered off and back on to reset the clock once an hour to still have accurate information.

A less significant issue that popped up during testing is the discrepancies between individual tests. This was not an actual issue in either the code or the nerf gun, but just a reality of how nerf guns function. The range of measurements was much larger than expected, which

caused us to question if our code or physical setup was functioning properly. In reality, the velocities of nerf darts vary greatly on an individual basis. For anyone using nerf guns in future projects, it is important to note that the velocity of the darts will be inconsistent and they are not a reliable source if looking to achieve a certain velocity every time.

The final issue we learned while developing this was timing with 3D printers. With all members being new to the practice of 3D printing, many issues needed to be solved before a viable casing could be printed. Receiving the 3D printing training during the MECH 324 (Dynamics of Machines) lab helped me understand the general process of translating a SolidWorks file to the printer itself, but the project-specific issues still posed a challenge. The first and most significant challenge of the printing process was honing down the print to realistically fit the project timeline. The casing was printed as eight different pieces, but only four could fit on the printing bed of a PRUSA MK3. With two separate prints, it was essential to have them print as efficiently as possible. Using PRUSA slicer to slice the files, the original print time was around 13.5 days. This was far too long and would have severely disrupted our testing schedule. The first change made to reduce print time was to change the print settings to print at 0.2mm SPEED, rather than the default .05mm ULTRA DETAIL setting. This reduced the print time to around 3.5 days. The final revision that enabled the print to fit the schedule was to change the infill pattern. The infill of a 3D printed product is the internal structure that provides support to the outer walls of the print. The traditional infill pattern is Grid, which is a rectangular grid structure that spans the interior of the object. The infill is pictured in dark red below.

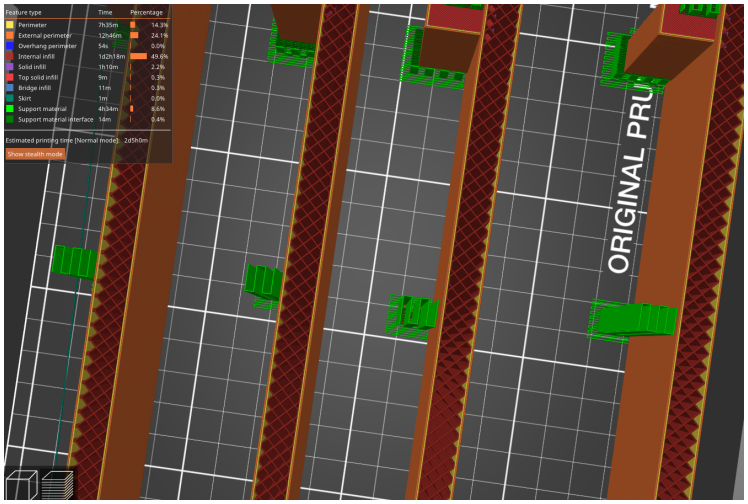

Figure 3: Grid infill on the Top Plates of the casing

The image above shows the print file for the four top plates of the casing. With the grid infill, the print time is still over two days. However, changing the infill from the grid to Adaptive Cubic resulted in a large reduction in print time, down to just over one day and 12 hours. The image below shows the cubic infill of the same file.

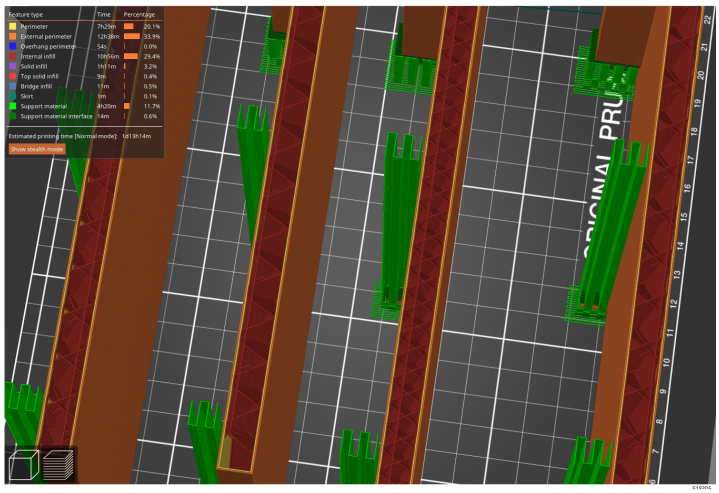

Figure 4: Adaptive Cubic infill on the Top Plates of the casing

The cubic infill still provides the support needed to create a solid casing, but the reduction of material reduced the print time as well. This is also related to the percentage infill,

which was very important to our print. As the casing will house a light-sensitive system, one objective is to limit the amount of uncontrolled light that enters the system. Ideally, only the light from the chosen LEDs will be in the system boundaries, so that changes in other lights do not impact system readings. That being said, it was decided that a 30% infill with black PLA filament would be sufficient in blocking out unwanted light. A reduction of infill would greatly reduce print time but would impact the ability of the system to block out light. It was decided to remain with a 30% infill to ensure the highest quality.

Even after a successful print, there are still changes that could be made to improve the efficiency of the print. The housing printed well and was completely functional. However, the long print time caused the bottom surface to mildly warp due to prolonged exposure to the heated bed. While this didn't impact our design, it is important to be aware of when printing something that requires higher levels of precision. Reducing the print time and printing at a lower bed temperature can help reduce warping for future prints. There were also minor issues with printing tall pieces with such narrow bases. As the bottom of the print solidifies, it can lose its adhesion to the printing bed (especially if minor warping has occurred). In this project, additional adhesive was added to ensure the parts stayed upright. To solve this problem in future prints, a "base" can be added. A base is a software-generated platform consisting of supports around the bottom layers of the printed object. A base increases the area of adhesion at the bottom of the printed object, increasing stability and lowering the chance that the piece will tip over during the print.

My recommendation for future students developing this project would be to try a faster sampling rate and a variable type that would allow more data to be stored. This would help with some of the overflow issues we ran into and the faster refresh rate would allow for a faster and

more accurate read of the object moving through the device. The last would be to have a more consistent device to propel your object through the gates. We were using a relatively cheap nerf gun and saw a lot of variation in the speed of the nerf dart passing through the gates. This makes sense as there are a lot of different variables that would affect a nerf darts performance such as dart weight, shape, size, spring force, wind, etc. In more ideal conditions with a better propulsion system, I believe this device would be very accurate.

Appendix

Detailed Wiring Diagram:

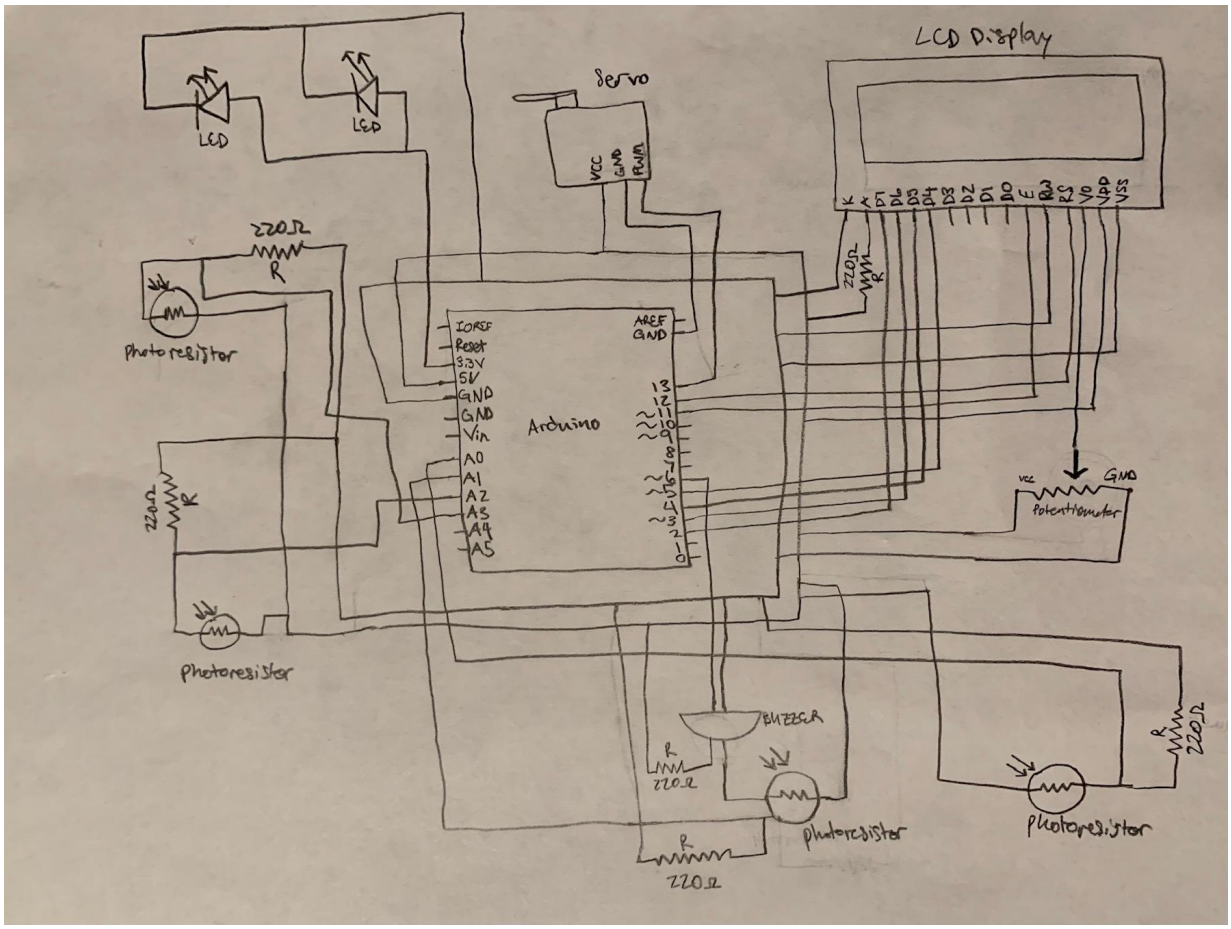

Figure 5: Functional Wiring Diagram

Functional Wiring Diagram:

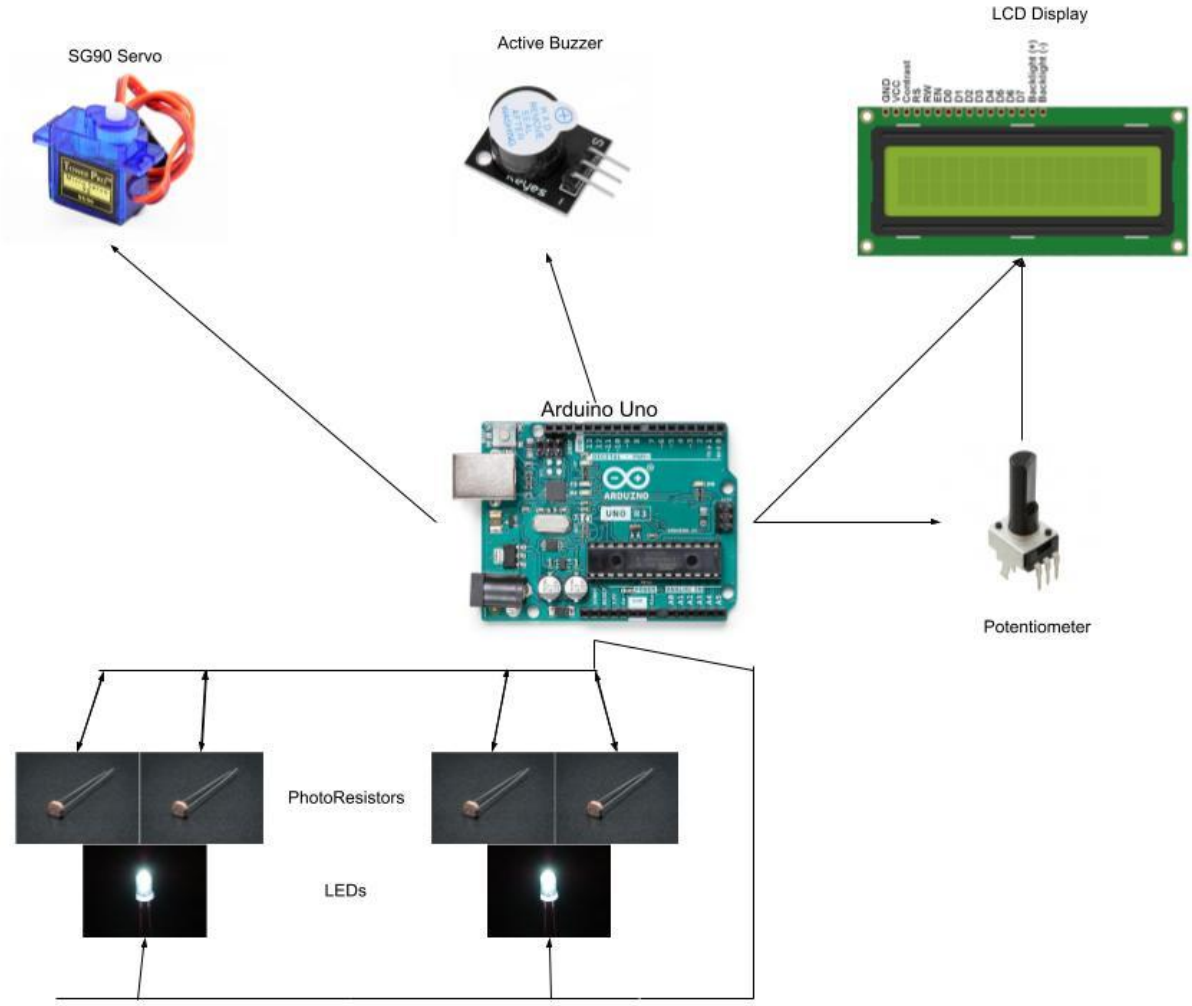

Figure 6: Functional Wiring Diagram

Code:

```
#include <Servo.h>
#include <LiquidCrystal.h>
int sensorValue1 = 0;
int sensorValue2 = 0;
int sensorValue3 = 0;
int sensorValue4 = 0;
int sensorAvg1 = 0;
int sensorAvg2 = 0;
int sensorAvg = 0;
long timet;
long gate2;
long gate1;
int pos = 0;

Servo myservo;
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup()
{
  lcd.begin(16, 2);
  lcd.print("Object Speed: ");
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  pinMode(A2, INPUT);
  pinMode(A3, INPUT);
  pinMode(6, OUTPUT);
  myservo.attach(13);
  Serial.begin(115200);
}


void loop()
{
  analogWrite(9,LOW);
  sensorValue1 = analogRead(A0);
  sensorValue2 = analogRead(A1);
  sensorValue3 = analogRead(A2);
  sensorValue4 = analogRead(A3);
  sensorAvg1 = (sensorValue1+sensorValue2)/2;
```

```
  sensorAvg2 = (sensorValue3+sensorValue4)/2;
  sensorAvg = (sensorAvg1+sensorAvg2)/2;
  //Serial.println(sensorAvg1);
  if (sensorAvg2 < 750)
   {
    gate1 = micros();
    if (sensorAvg1 < 250)
      {
      timet = micros() - gate1;
      lcd.setCursor(0, 1);
      lcd.print(0.3UL/(timet/1000000UL));
      //delay(1000);
      lcd.print(" ");
      //activate buzzer to alert user it triggered
      digitalWrite(6,HIGH);
      delay(1000);
      //reset buzzer to off to not annoy the user
      digitalWrite(6,LOW);
      //if the object speed is less than 10 m/s set flag to pos 60
      if (timet <= 10)
      {
        myservo.write(60);
        delay(700);
        myservo.write(0);
      }
      //if the object speed is less than 10 m/s set flag to pos 120
      if (timet <= 12 && timet > 10)
      {
        myservo.write(120);
        delay(700);
        myservo.write(0);
      }
      //if the object speed is less than 10 m/s set flag to pos 180
      if (timet > 12)
      {
        myservo.write(180);
        delay(700);
        myservo.write(0);
      }
      }
    }
}
```