Shooter Game

CS122A: Fall 2017

Anthony De Belen

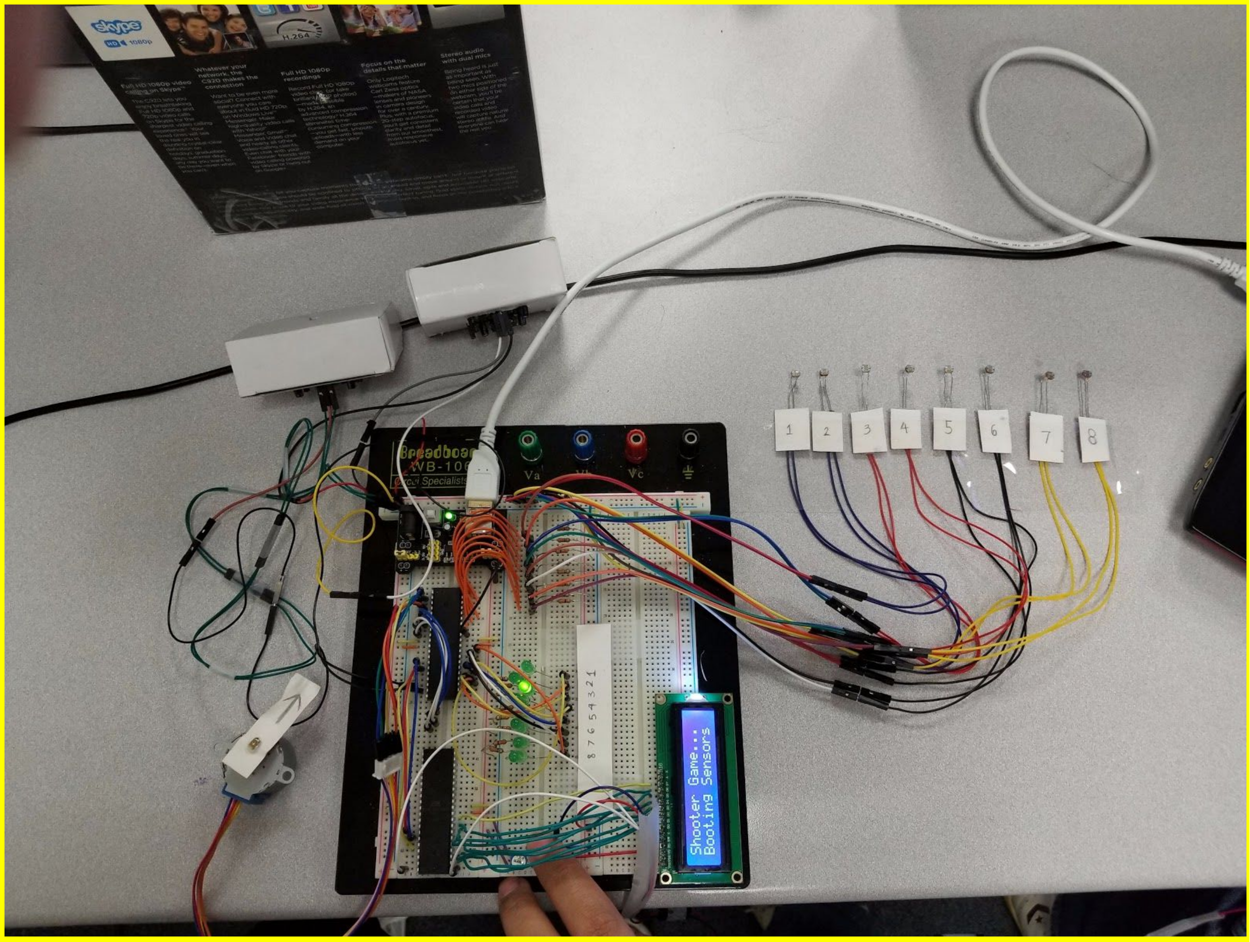
Table of Contents

Introduction

I have made an open world RPG for CS120B. It is very linear with basic controls that most games have today. But my passion for games does not stop there specially with the new technology available today. I have been amazed by virtual reality and how it will change gaming in general. Therefore my attempt will be to make a game that does not use conventional controllers but motion capture as inputs for the game.

I will be making a shooter game with the use of a trigger enabled laser connected to a stepper motor. The game will have eight different targets which can be hit when the stepper motor rotates clockwise or counterclockwise. Because there are eight targets, the motor will have to rotate 45 degrees. The motor will be controlled using two motion sensors for a left and right gesture. A LCD display will display the score each time the correct target is hit.

This is mainly a form of entertainment and an attempt for something unique. The purpose of the motion sensor is to get input from the user by detecting any kind of motion either from the left or right sensor. The game will be accumulating as much points as possible in given amount of time which is the same environment as arcade games.



Hardware

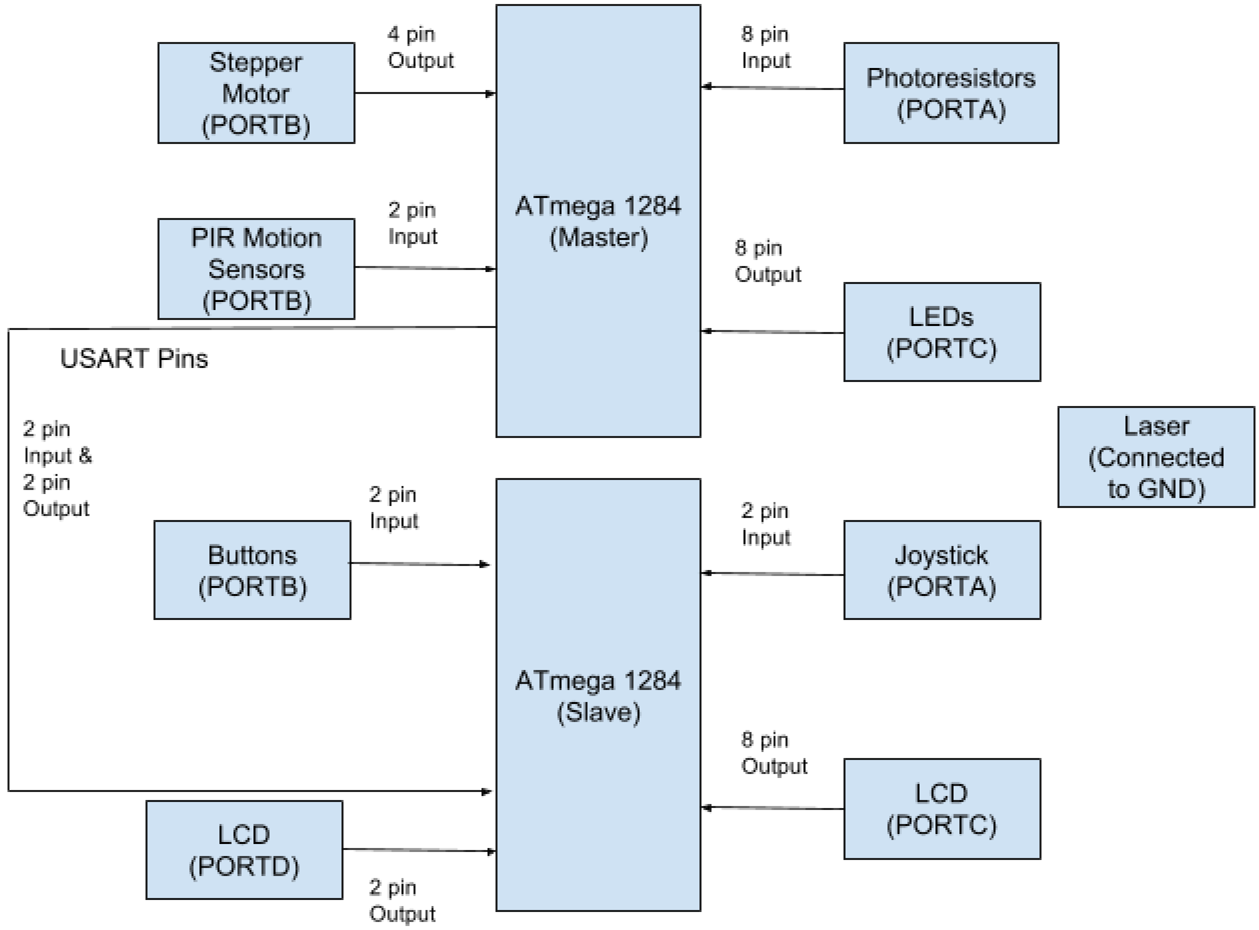
Parts List

The hardware that was **used** in this project is listed below. The equipment that was not taught in this course has been bolded. *Include part numbers when available*.

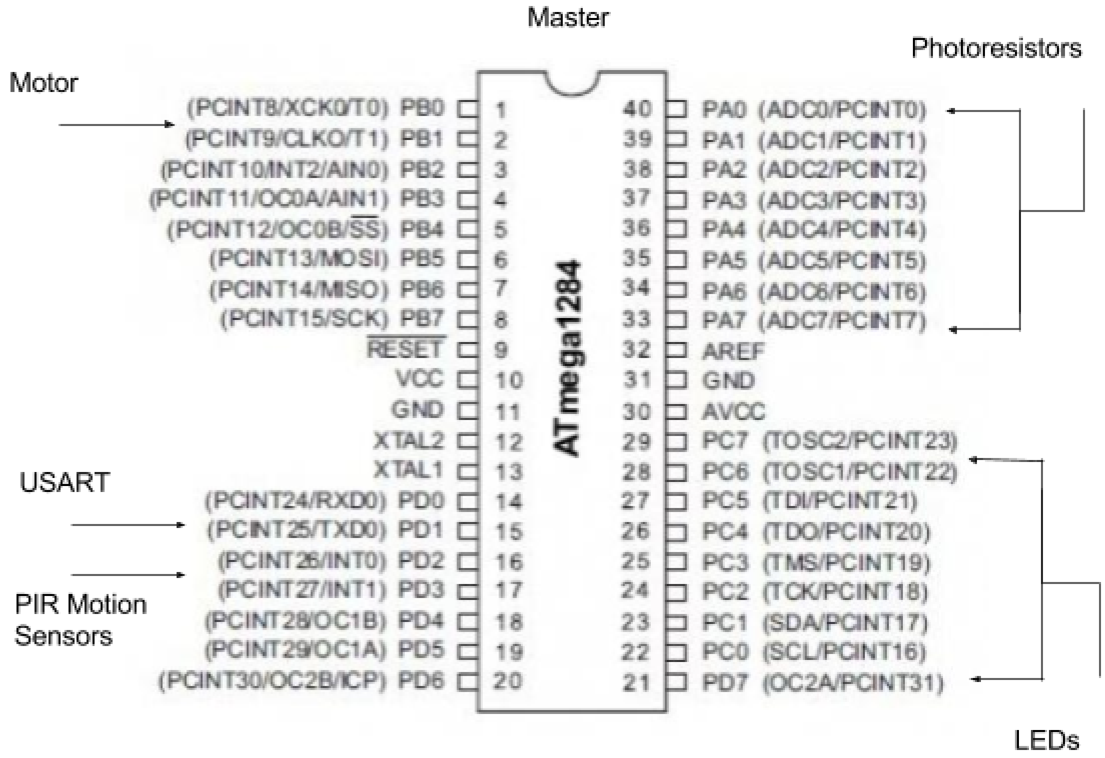
Part	Part #	Quantity	Price (optional)
ATMega1284	ATMega1284	2	
PIR Motion Sensor	**HC-SR501**	**2**	**$4.40**
Photoresistors	GM55 Series Photoresistors	8	$2.40
LEDs		8	

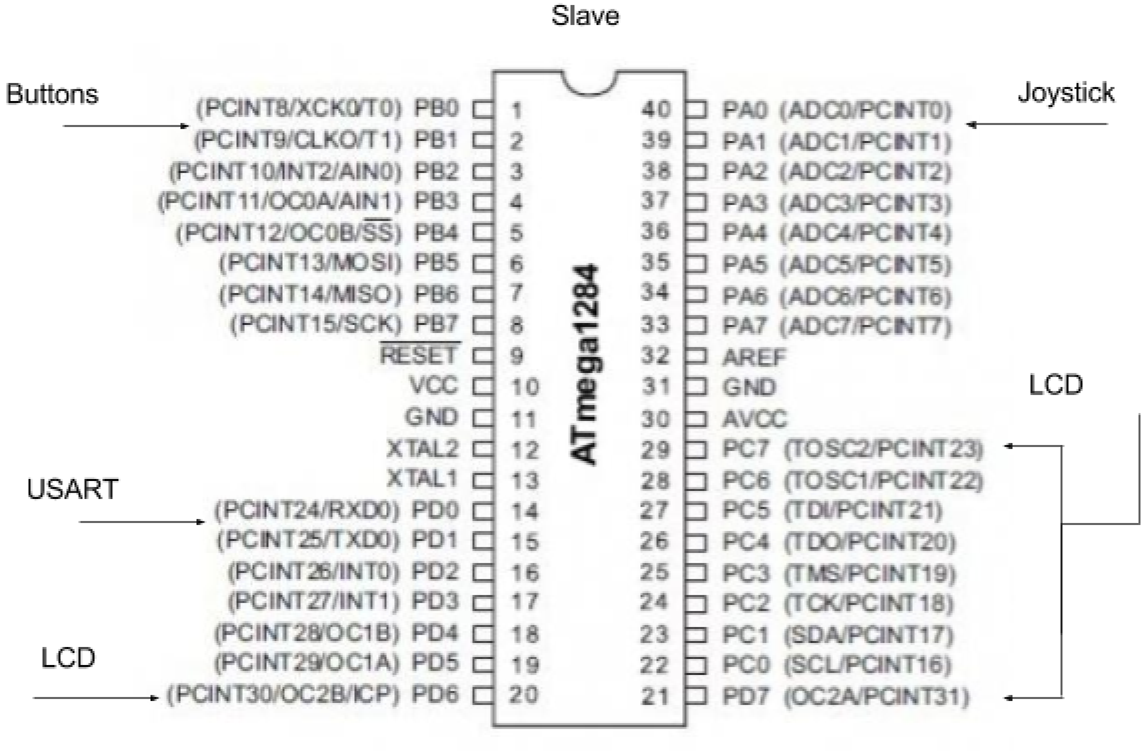
LCD Display		1	
Joystick		1	$5.00
Dot Diode Laser	**80-141-10**	**1**	**$3.00**
		Total	$14.80

Block Diagram



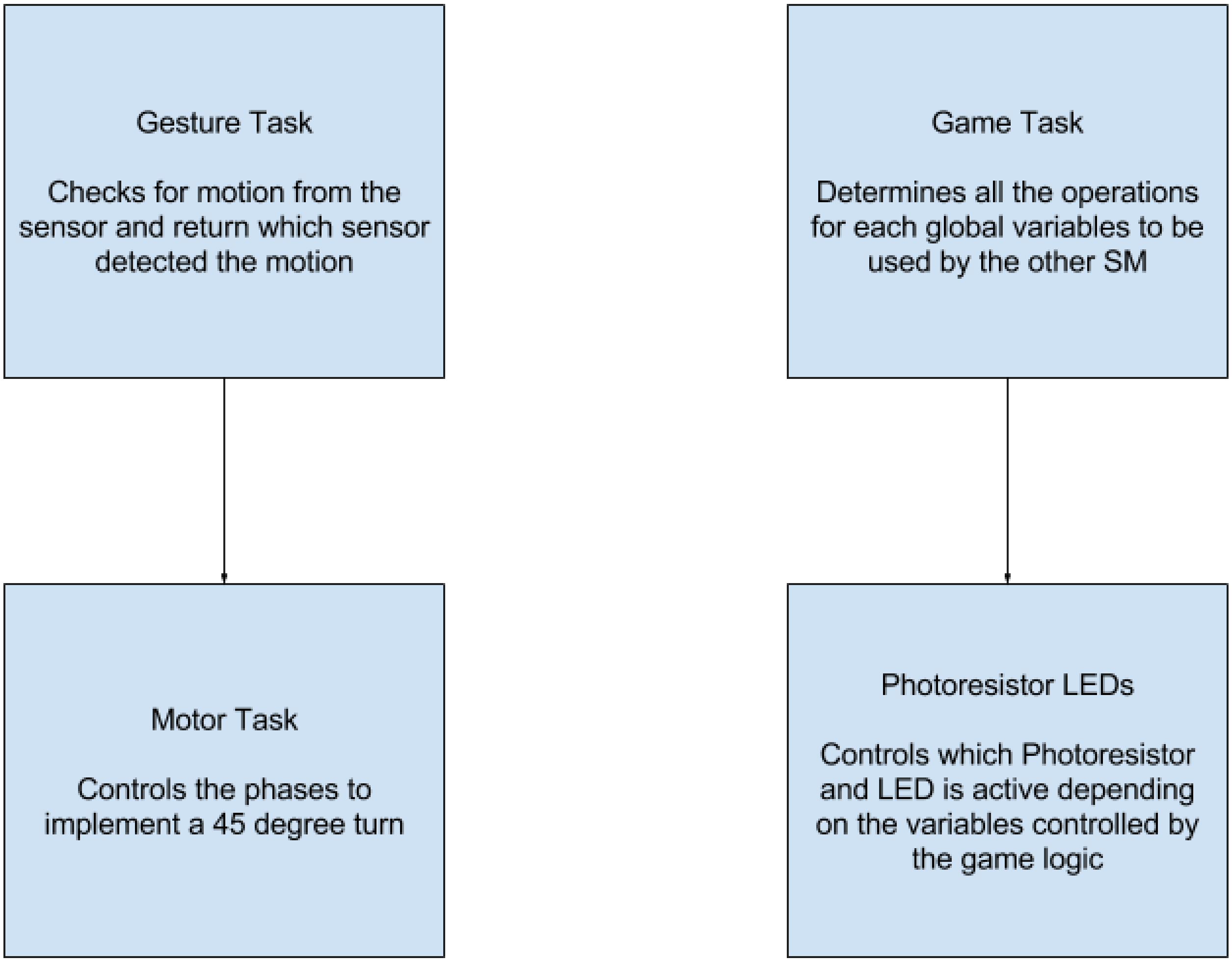
Pinout (For each microcontroller/processor)





Software

The software designed for this project was implemented using the PES standard. The overall design as a task diagram is included below.



Gesture Task

Checks for motion from the sensor and return which sensor detected the motion

Game Task

Determines all the operations for each global variables to be used by the other SM

Motor Task

Controls the phases to implement a 45 degree turn

Photoresistor LEDs

Controls which Photoresistor and LED is active depending on the variables controlled by the game logic

Gesture Task:

This task is connected to two inputs which are the two motion sensors. If a motion is detected, it updates a variable which we will use to determine the direction of the motion when it turns.

Motor Task:

This task accepts the input from Gesture and determines whether it will do a clockwise or counterclockwise direction. The phases are already determined from the elective lab and the only thing that's changed is having the motion turn for 45 degrees.

Game Task:

This task determines which is the current target and if determines if the current target is hit by the laser. This task will output the photoresistor and led codes that will be used by the other task.

Photoresistor LEDs Task:

This task will output the active LEDs corresponding to the right photoresistor target and initializes the correct ADC pin for photoresistor. This will keep updating depending on the Game Task.

Implementation Reflection

At this point, the project feels incomplete due to some necessary functions being omitted for example, a reset button. This is fairly easy to implement but I mainly focused on the newer ideas that I need to implement with the focus of getting the game logic set. The process went a little off from how I saw it to unfold due to complications in the hardware. Overall, the planning was poorly done, which resulted to a huge delay for the milestone.

If given a second chance to redo the project, the first thing to do is research the parts that will be used. For example, gesture sensors can be bought from a manufacturer or built by yourself. It is also bad to assume that all gesture sensors will work on every microcontroller without the proper specification reading. Another is looking at the big picture and not just the code. This is one of my mistakes because I did not spend much time thinking about the final design of my project. I had a hard time making a base/container for my project which made it really hard to demonstrate to the graders.

This project was really interesting because even though it is simple, the game is still unique when compared to similar games in its genre. With the advance in technology, even games will evolve to introduce new forms of stimulation. Therefore, this is the reason why I tried to imitate what input controls from virtual reality games to an old fashioned controller based game.

Milestone

The target milestone is to get gestures as a form of input and use this input to move the stepper motor accordingly. We have never used sensors before that’s why making driver for the sensors will be most of the new ideas that I will have to implement. I was able to demo my milestone at the start of week 8 because of my lack of research on the gesture sensor. Switching parts is the easiest option and after getting permission, I had to wait for the new parts to arrive. This led to the lack to time to fully implement everything.

Because the original gesture sensor uses I2C communication, I had to change parts that I can easily implement on the atmega1284.

After getting the approval to switch parts, I switched to PIR motion sensors which I easily implemented and working perfectly. However, this hardware have multiple flaws. One is that it takes 1 minute to boot the sensors or else it will give incorrect values

Completed components

I completed up to 80-90 of my proposed project with reason stated on the milestone section. Because the original gesture sensor uses I2C communication, I had to change parts that I can easily implement on the atmega1284.

After getting the approval to switch parts, I switched to PIR motion sensors which I easily implemented and working perfectly. However, this hardware have multiple flaws. One is that it takes 1 minute to boot the sensors or else it will give incorrect values. And two, PIR motion sensors have different block timings where it does not accept input (first sensor had a minimum of 3s, second was supposedly 0.3s but it appears to have around 1.5-2s block timing) until a specific amount of time. Getting this to work was my 70-80.

My 80-90 is getting a completely working game with a game logic and score system. I was able to demo this on time but because of the lack of time, I was not able to complete my 90-100.

Incomplete components

The components that I did not complete are mostly parts that we have used before in embedded systems classes. For example, adding buttons for reset which instead replaced with a game that continues to replay until the machine is turned off. Another would be adding joystick as another form of input for the game. This was also done as an elective lab and the idea of using multiple ADC pins was also implemented for my photoresistors.

Some other game logic that I have not implemented are the magazine count for the laser so players cannot repeatedly trigger the laser or have infinite ammo. Also having multiple modes for the games to add more diversity to the form of entertainment it can offer. As mentioned above, I spent so much time configuring and replacing my form of input for my game that is why I was unable to complete these parts.

Youtube Links

- Short video:
    - https://www.youtube.com/watch?v=Y61f37aIkZ4

Testing

Motor

To test the motor, I made sure to have the motor only rotate for 45 degrees. I let my TA Mark do the final check to see if the motor does rotate for 45 degrees even when a constant value was being inputted. Some test cases would be if the 45 degrees turn was complete but the the sensor still detects motion, therefore the motor will continue to move until it finishes another 45 degree turn.

Photoresistors

Each of the photoresistors will be linked to a specific ADC pin, therefore, the initial test is to see that I am getting the correct value for each photoresistor. My lab partner tested out each photoresistor using LEDs to see if the correct LED lights up with the corresponding ADC pin. Each photoresistor that is linked to one led lights up when we shine the laser on it.

PIR Motion Sensor

Testing and calibrating the motion sensor required a lot of time. We had to make the sensitivity and timing set to the lowest or It'll detect even the slightest motion. Therefore we put each motion sensor in a box to limit its detection angle from 100 degrees to somewhere below 5 degrees. After that I have multiple peers test whether they can have the sensors detect something from far away. Eventually, I got a box with a really small hole, and it only detects motion that are a couple of inches away from the hole.

Game Logic

To test the game logic, I used the Atmel Studio simulator to check the values of each of my variable as they get updated in game. I have a lot of global variables that also gets sent to the other microcontroller therefore I have to make sure I have the correct value every time. Through the simulation, I have easily fixed problems like input weird characters on the LCD display and resetting values after the game is done.

Known Bugs

- Rotation bug

The rotation appears to be a little off by around 1-2 degrees. This is a small difference but it is noticeable in the long run. I believe the cause for this is the motion sensor which has a timing block of around 2 seconds, therefore as soon as one 45 rotation is done, it still continues to rotate for a bit which causes the bug described above. The only fix for this i believe is to get a better motion sensor that actually has little to none block timing.

Resume/Curriculum Vitae (CV) Blurb

This project can be transformed to how you want it to look. It can be shooting aliens or putting out fires with just a small change of parts but I am calling this project, “Attack on Aliens”. In this project, sensors were used to simulate gestures to control a not so harmful type of gun namely a laser. The laser will be lined up with the photoresistors to score a kill. I used the functionality of atMEGA1284’s multiple ADC pins to include multiple targets and read each appropriate photoresistors. The gun is attached to a stepper motor which moves 45 degrees on the desired direction(left of right).

Future work

Attack on Aliens is far from being perfect. To make the game even harder, the sensors needs to be upgraded to have no block timing to change the mode of input to be continuous. The motor needs to be replaced as well to have more RPM because the game feels really slow. This two new parts will make the game’s response better and smoother. Instead of doing 45 degree turns, we can have a free for all 360 degree freedom when it comes to targeting.

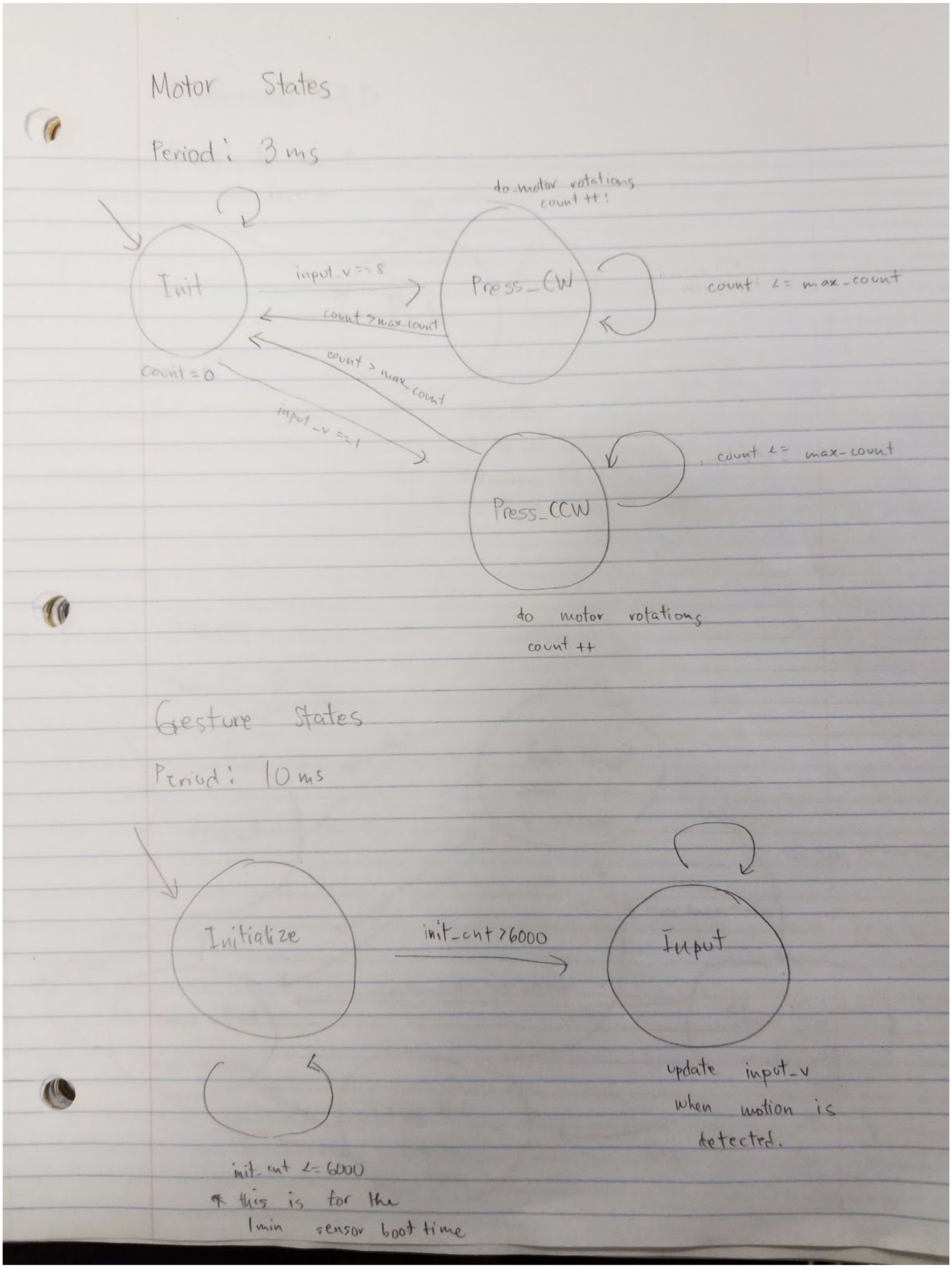
This game definitely needs a dome shaped case where the lower half will house the motor and microcontrollers which should be hidden from the player. While the dome will be made of glass/plastic where the photoresistors, leds and laser is located. A theme needs to be included in the design of the case to show more creativity.

My company research report focuses on virtual reality. The completed version of this project can be used to apply to companies that focus on VR because of how inconvenient these machines are (for example, HTC Vive’s sensors must be setup in a preferably large room) in terms of space. A better controller for HTC Vive might have some connection to my project.

References

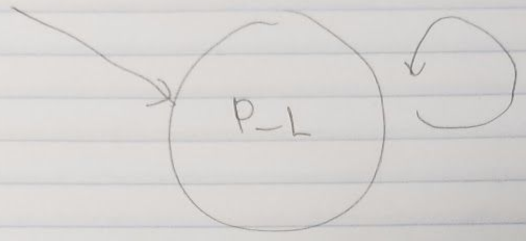
I have used to code as a basis to write my PIR motion sensor code. Note: His/Her code does not work but I rewrote it and got it to work. https://pastebin.com/R8XhNKGs

Appendix

Motor States
Period: 3 ms
Init
count = 0
input_v == 8
Press_CW
do_motor rotations
count ++!
count <= max_count
count > max_count
count > max_count
input_v == 1
Press_CCW
count <= max_count
do motor rotations
count ++
Gesture States
Period: 10 ms
Initialize
init_cnt >6000
Input
update input_v
when motion is
detected.
init_cnt <= 6000
* this is for the
1min sensor boot time

Photoresistor LED

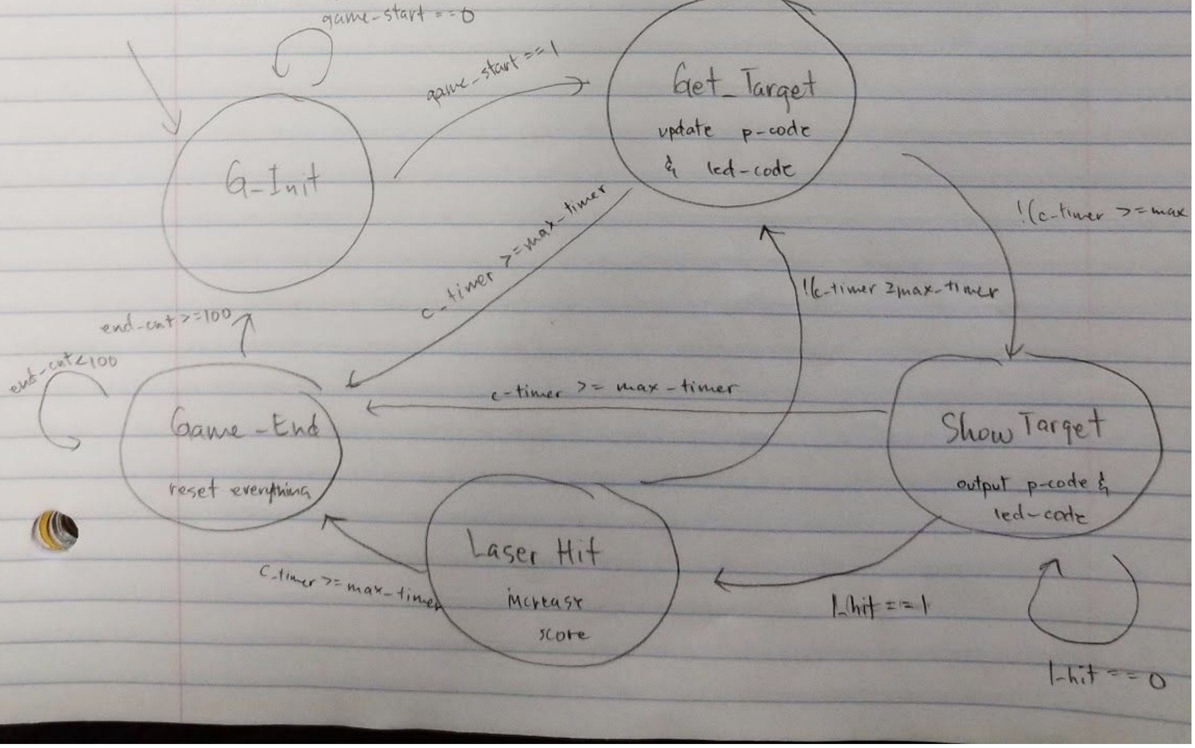
Period: 10 ms



set the ADC Pin
and set the led code
also determines the ADC value and
check if it is a hit or not.

Game

Period: 50 ms


game-start == 0
game-start == 1
G_Init
Get_Target
update p-code
& led-code
!(c-timer >= max
c-timer >= max_timer
!(c-timer ≥ max-timer
end-cnt >= 100
end-cnt < 100
Game-End
reset everything
c-timer >= max-timer
Show Target
output p-code &
led-code
Laser Hit
increase
score
C_timer >= max_timer
l-hit == 1
l-hit == 0