FAMU-FSU Engineering
Senior Design 301
2021-2022
Operation Manual

I. Project Overview

Our project was the IEEE SoutheastCon Hardware Competition where we were tasked with creating an autonomous robot that could complete various tasks throughout a course. We had to collect beads off the “trees” and throw them into nets as well as follow a path and push a marshmallow out of the way.

II. Module Description

A. Driving Base

Our robot integrated an Arduino Mega with 7 line sensors and a Motor Driver to control the Motors. It worked by taking in the values of the line sensors and in turn sending the information to the Motor driver to keep the robot centered on the line in the path.

Pseudocode:

```
Function LineTrack
LineData [*Array of Data*];
Lookahead
Current position
Angular velocity
    LineData = GetData

Case (LINE)
If line (ones in the array) not in center of Lookahead
    Find current position
Find distance of lookahead & current position
    Find angle from center
    Calculate Angular Velocity
Correct the path trajectory DriveTrain
```

B. Image Recognition

Since there were going to be nets or cups to deposit the collected beads into, we needed an image recognition software to communicate with the top Arduino Mega when to shoot the beads into the nets. We used a PixyCam to recognize the Red Solo cups to let it know not to shoot and when it didn’t see the red Solo cup it would shoot. Since we couldn’t train the PixyCam to see the white net with the white background and the nets/cups had fixed potential locations, this approach worked well. The PixyCam was built to be used with the Arduino MEGA and has a specialized set of pins dedicated to it on the arduino. Training is done by a push of a button on the camera.

C. Bead Collector/Thrower

In order to collect the Beads off the trees, we used the top Arduino Mega with a 3D printed arm that had 3 servos to give the arm enough rotation and movement to be able to collect from the trees and deposit them into the 3D printed spring-loaded catapult. When it is ready to launch, the arm will hold the catapult down while a servo motor pulls the spring to tension and when the arm moves it will launch into the net.

Pseudocode:

```
Pins for Servos
Pins for PixyCam
Pins for communication

If (pick up beads)
        Beads picked = 1
        collect();
If (launch beads and pixycam sees cup)
        Tell robot to keep moving
Else If (launch beads and Beads picked = 1)
                Tell robot to keep moving
launch();
Beads picked = 0
else
        Tell robot to keep moving

collect()
    Extend arm out and grab bead
        Deposit bead in to catapult
        Return arm to rest position

launch()
Verify Net or Cup
Move arm to block catapult
        Tension catapult
        Move arm to release catapult
        WAIT
        Release catapult to rest position
        Return arm to rest position
                                End function
```

D. Marshmallow Mechanism

The robot used a front bumper to passively push the marshmallow into the alleyway since it wasn't worth that many points. We also added cardboard to the sides in front of the wheels since the marshmallow seemed to get stuck between the wheels and bumpers during practice runs.

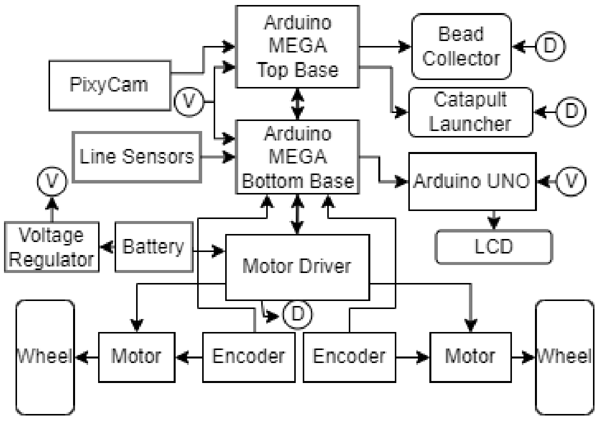
E. Light Up Display

The light-up display is an SD card LCD screen powered and triggered by the Arduino UNO.

Pseudocode:

```
Include Core Graphics Library
Include Image-reading functions
Loop(Read SD card){
Print to screen;
Delay;}
```

III. Integration

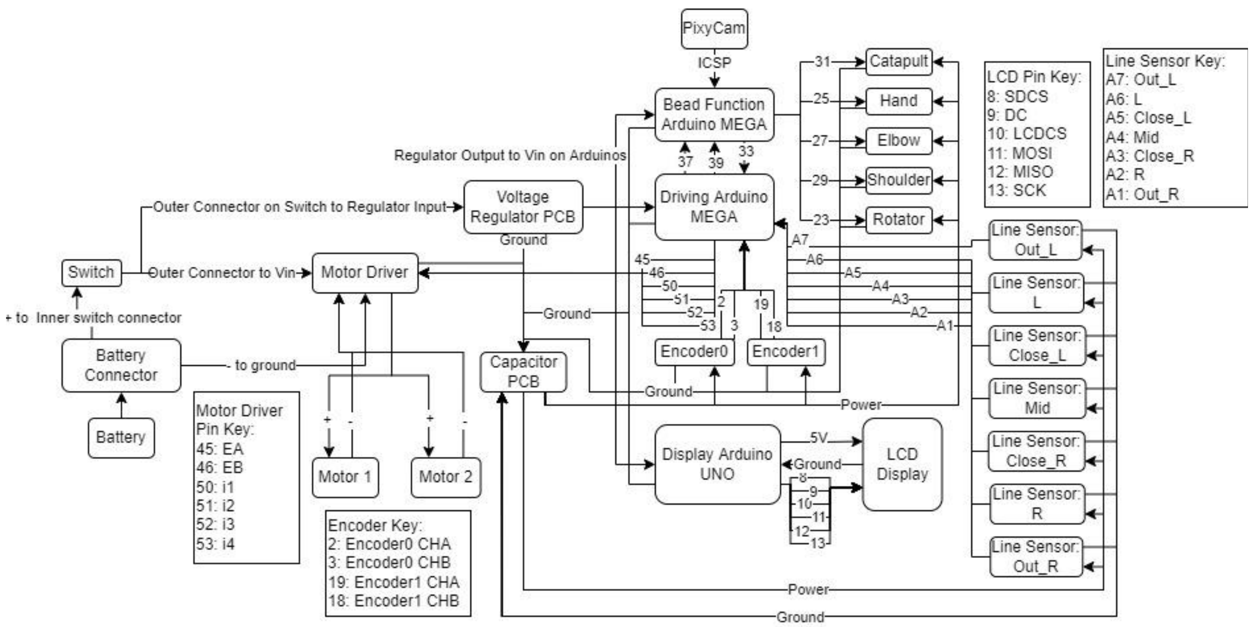


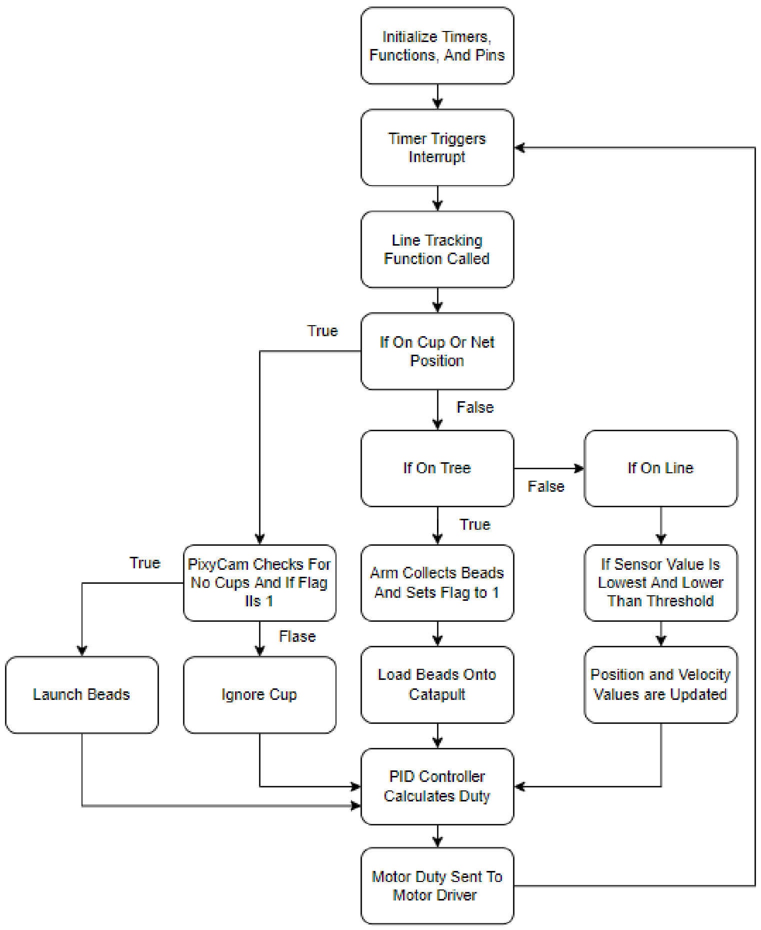
The diagram above is an overview of the connections between the modules on the robot. The robot consists of two tiers: the top base and the bottom base. Each tier has its own general purpose, with the bottom base serving mainly as a navigational and power module and the top base serving as a functions module, with the image recognition, catapult, and the arm.

The bottom base is made up of the navigation components and marshmallow mechanism. The navigation is accomplished by connecting the motors to the encoders and motor drivers and having those components communicate through the bottom base Arduino MEGA. The Arduino is also connected to the line sensors which collect data that informs the navigational algorithm to compute its next decision. That decision is then used to drive the robot forward through a PID controller that sends data to the motor driver and drives the motors forward. The marshmallow mechanism is a passive module that integrates straight into the front of the bottom base, offering a way to attempt the marshmallow pushing task, as well as a mounting place for the line sensors to get good readings for the navigation.

The top base of the robot consists of the rest of the modules that are required for the robot's successful operation. These modules are the catapult, arm, and the PixyCam. These modules all communicate with each other through the top base Arduino MEGA via data pins. They are programmed to be synchronous with each other in order to perform the necessary tasks. To prevent failures during the operation of these modules, servos are turned off when not in use. The final module in the top base of the robot is the screen which just uses simple instructions from the Arduino UNO to load an image from its SD Card.

The integration of the top and bottom base starts with the need to share power from the battery and ends with how they communicate with each other. All of the motors (DC and servo) require large amounts of the current, so they are powered separately from the Arduino microprocessors and sensors: through the motor driver. This is to prevent brownouts from resetting the microprocessors. The Arduino on the top and bottom base are powered from the battery through a 5-volt regulator. The way the Arduinos communicate is by sending binary signals through jumper wires from port to port with encoded messages. Included below are a detailed wiring diagram and code flow-chart.




Initialize Timers, Functions, And Pins
Timer Triggers Interrupt
Line Tracking Function Called
If On Cup Or Net Position
True
False
If On Tree
False
If On Line
True
PixyCam Checks For No Cups And If Flag IIs 1
True
Flase
Arm Collects Beads And Sets Flag to 1
If Sensor Value Is Lowest And Lower Than Threshold
Launch Beads
Ignore Cup
Load Beads Onto Catapult
Position and Velocity Values are Updated
PID Controller Calculates Duty
Motor Duty Sent To Motor Driver

IV. Code Repository

GitHub was utilized for source control for our project. The repository name is SD301-southeastcon (https://github.com/kelvham/SD301-southeastcon) and was created on 1/7/2022. We only used one branch, main, and did not use the “issues” or “pull request” features, as we had other means of communication and were highly collaborative. 126 commits were made that included test code and inevitably scrapped components. The unused files are no longer in the current version of the code, but can be found in previous commits. Feel free to fork the repository for further use/changes, or simply download the .zip file under release v1.0.0. Do not make any pull requests or issues, as the repository is no longer being maintained. Since there were three Arduino’s used on the robot, there are three programs required to run the robot. For the display running off the Arduino UNO, the file is named *Display_UNO.ino*, which is located in the Display_UNO folder. Please note that in order for the display to output the image correctly, the inserted micro-SD card must have the image loaded with the filename “logos.bmp”. The file that we used can be found in the same folder. For the motors and line sensors running off of one Arduino MEGA, the file is named *RobotBaseCode.ino*, which is located in the RobotBaseCode folder. Note that this file is dependent on two .h files (*LineTrack.h* and *RobotBaseCode.h*), which are both in the same folder. For the catapult and arm running off of the other Arduino MEGA, the file is named *LaunchCatapult.ino*, which is located in the LaunchCatapult folder.

V. Operation

To operate the robot, place it on the starting square on the track it was designed for. Then flip the power switch to the on position and let the robot begin its autonomous operations. Once it is done with its tasks, it will stop and it can be turned off by flipping the same power switch as before. Because of inaccuracies with the positional data, the robot will occasionally get off track and fail to follow the line and collide with the barricades. When this happens, there is a low chance of recovering and the robot should be powered off to prevent burning out the motors.

VI. Troubleshooting

1. There may be issues with the line sensor readings which can be monitored by printing the values using UART and adjusting the values in the code to account for the variation.
2. If the robot is constantly shooting no matter what is in front of it, verify the lens cap isn't on.
3. When it is turned on and some components aren't turning on or aren't working properly, verify the wiring is correct. Since jumper wires were used, they often come loose in transit.