EXECUTIVE SUMMARY

Team 8 was tasked with building a robot to prepare for human exploration of mars. This meant building a robot that could collect scatter tennis balls, transport the balls to a storage location, and dispense the balls into the storage container. The team took on the challenge by designing and building a chassis that had the ability to “engulf” up to three tennis balls. The movement and identification of objects was controlled by implementing an FSM with the feedback of a Color-Sensitive Vision Sensor. Container and robot communication was enabled by an IR sensors coupled with infrared indicators. Finally balls were dispensed with use of both propulsion and a push plate mechanism. The team was successful and was the victor of the final challenge round. The team’s learnings were related to the sensitivity of the Pixy camera, the design of the PCB, and the reliability of IR sensors in a hot environment where there are many people. Future recommendations made by the team are to consider a more robust vision sensor and to consider a different method of remote communication between the container and the robot. Overall the team is satisfied with the design.

1. INTRODUCTION

In the following report, Team 8 discusses the mechanical systems of the robot and container, the electronic hardware that was selected for object detection, propulsion, and remote communication, and the software that was created to put it all together. Testing and results are shared in the report as well as the team’s learnings.

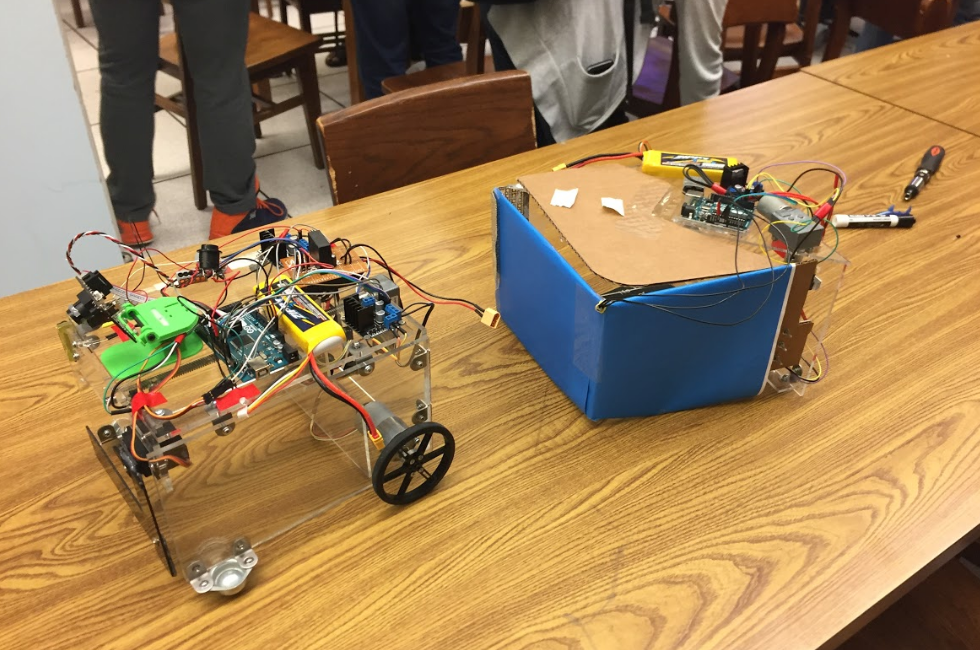


Figure 1 : Team 8’s Final Robot and Container

1. MECHANICAL SYSTEMS
   1. *Chassis and Component Mounts*

The robot chassis design was such that it would “engulf” the ball by driving over the ball and trapping it with use of a trap door at the front. The balls, once trapped in, would then move on the floor along with the robot. For this strategy, there was a need for an enclosed chassis with side and back walls. The design ensured there was enough space to collect and contain a minimum of three balls and allow space for the balls to move around freely within the enclosure. The chassis was made wide enough to avoid stringent alignment of the robot center with the ball while collecting.

The chassis design went through cardboard prototyping before the final chassis was laser-cut from a 0.25 inch thick acrylic sheet. Acrylic was chosen for its strength in bearing the load of the motors and batteries. It was the choice also because of its suitability for laser cutting. Laser cutting was selected for manufacturing as it eliminated the the need for precise hand drilling of holes for motor mountings and the L-brackets which were used as connectors and angle reinforcements between the chassis faces.

The driving motors (12V DC) were mounted on each of the side walls of the chassis at the rear end. Rear wheel drive was decided upon to provide maximum space for accommodating the balls. The driving wheels were mounted on the motor shafts. Two castor wheels were used as the front wheels of the chassis. The casters decreased the required start-up torque for situations when immediate direction change was required since the casters allow movement in any direction. Care was taken that the height of the motor mounting was such that the wheels and the castors aligned accordingly to give a horizontal base.

The robot design was rigid, robust and stable. The height to width ratio of the chassis was such that the load application did not topple the robot. The final dimensions of the chassis were 8in x 10in x 6in.

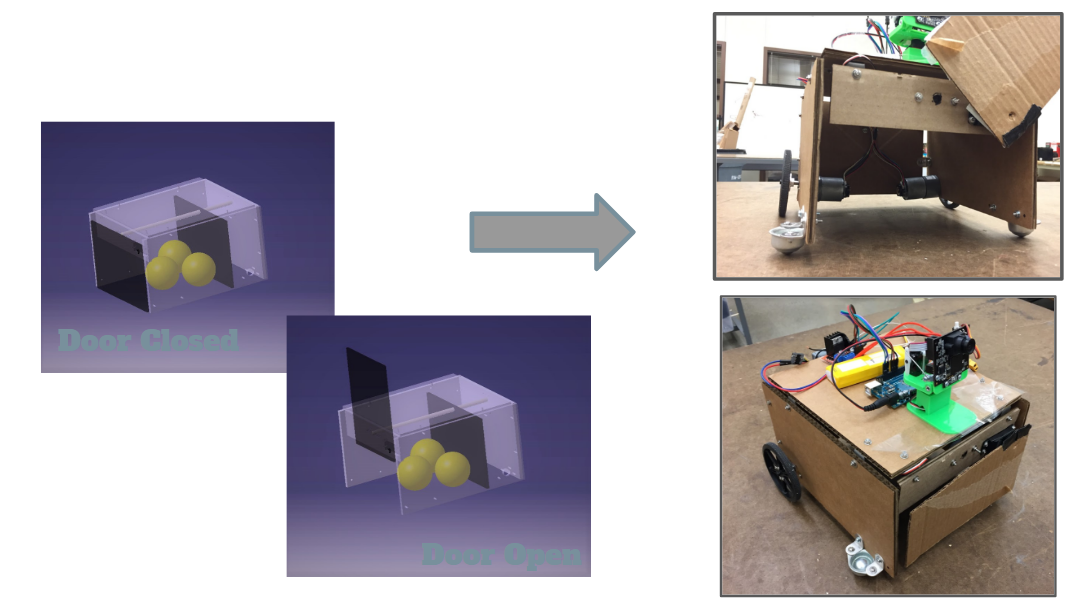


Figure 2 : Robot Design

* 1. *Trapdoor Mechanism*

The trapdoor was L-shaped with one end fixed to the servo motor shaft. The servo was mounted on the front face of the chassis. The door was closed in the default position and was opened by activating the servo, which rotated the shaft (by 90°) to open the “door”. The door was L-shaped to prevent it from blocking the view of the Pixy in the fully open position. It was made from sheets of carbon-fibre to keep it strong enough to resist any impact from the balls inside the robot and possibly giving way for them to escape. At the same time, the door was to be light to not induce heavy torque on the servo motor. Carbon fiber is expensive and stronger than what was needed for the door application but it was on hand and free to the team.

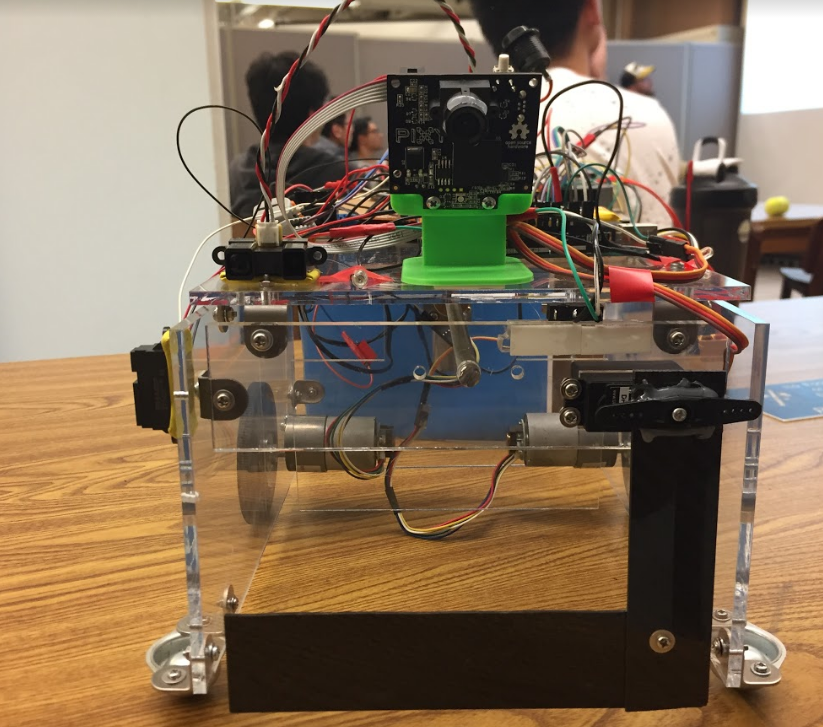


Figure 3 : View of the trap door

* 1. *Dispensing Mechanism*

The balls were dispensed into the container using a push-plate in the robot. A lead-screw and nut mechanism was used to drive the forward and backward motion of the push-plate. A 400mm long screw was chosen for the travel of the nut along the length of the robot. The lead screw was coupled to a 12V DC worm gear motor (high torque) which was mounted on the back wall of the chassis. The free end of the lead screw was supported by a pillow block bearing mounted to the front face of the chassis. The coupler between the screw and the motor shaft allowed for some amount of axis misalignment - although careful attention was made in the design to keep the axis perfectly straight.

As the lead screw rotated, the nut moved along the screw. Reversing the rotation of the motor reversed the motion of the nut. The push-plate was attached to the lead-screw nut using fasteners. The plate was made of acrylic for it to be strong enough to push the balls and bear the torque exerted by the screw. The dimensions of the plate were decided such that the rotation of the plate was prohibited by the physical constraints of the chassis side and top walls. A slight clearance was given to avoid friction between the surfaces. Limit switches were added on the chassis to limit the forward and reverse extent of travel of the nut on the screw.

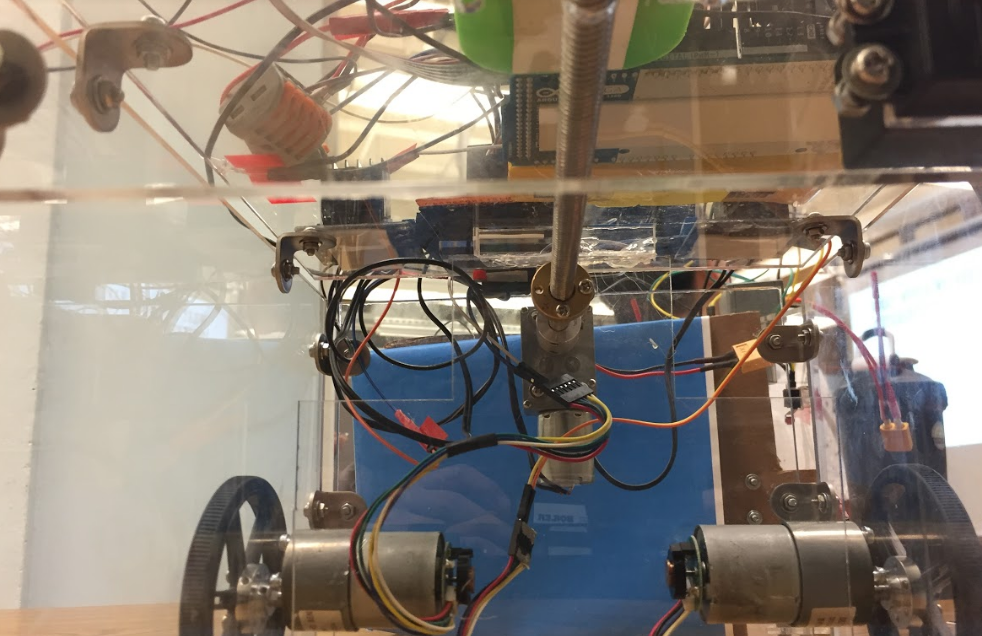


Figure 4 : View of lead screw mechanism and drive motor mounts

* 1. *Container*

The container was designed to be a triangular-base prism, with walls parallel to the arena walls (i.e., two walls at right angles) and a door covering the third face. The triangular design was selected so that the robot need not align perfectly with the door, and hence could approach the container in any orientation with respect to the door while dispensing. The door lifting mechanism used the same lead screw and nut mechanism as the robot but oriented in vertical direction. Part of the lead screw used in the robot was recycled for this use, it was cut and the remaining part of the screw was used. The lead screw was coupled to a 12V DC worm gear motor mounted on the roof of the container. The walls and roof of the container were laser-cut from acrylic sheets for precise placement of mounts. The door was made out of cardboard because the door was a cantilever supported at the screw end, hence it would have to be light to not cause considerable moment on the nut and screw.

A right angle bracket was attached to the lead-screw nut and the door to connect the two components. The free end of the screw was supported by a bearing mounted to a right angle bracket that was attached to the side wall of the container. As the screw rotated, the nut moved up and down along with the door. Since the free end of the door was not constrained physically, a guide was provided on the container guide the motion of the door. The guide was mounted at a height greater than the ball diameter so that it would not obstruct the incoming balls. Limit switches were mounted on the container to limit the extent of the nut travel on the lead-screw.

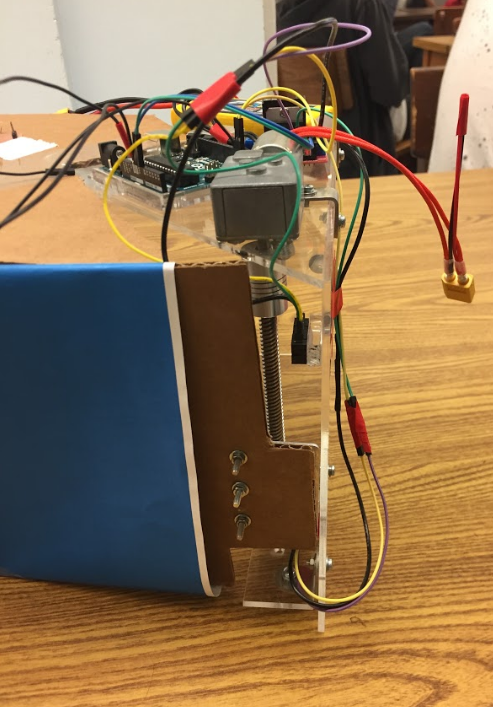
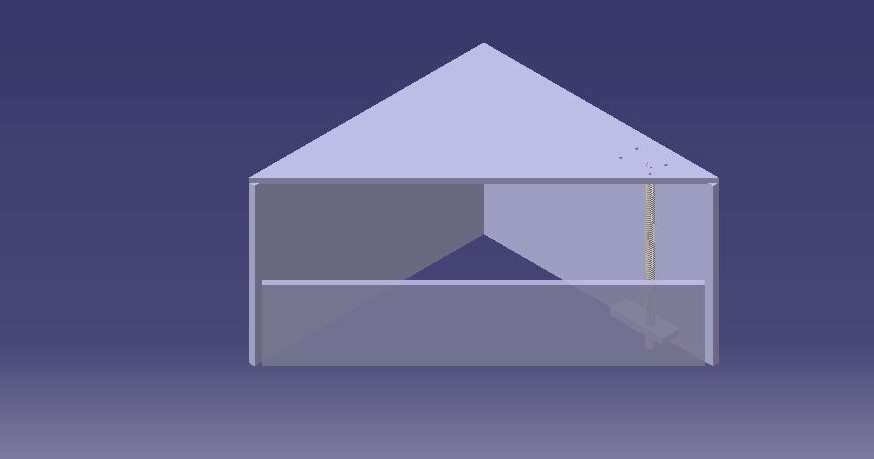


Figure 5 : Container design

1. ELECTRONICS AND POWER SYSTEMS
   1. Microcontroller

The team used an Arduino Mega 2560 as the controller on the robot because the number of pins accommodated the Pixy camera, three DC motors, a servo motor, two IR sensors, and an infrared emitter. For the container, an Arduino Uno was used as the controller since only one DC motor and a infrared receiver needed to be controlled.

* 1. PCB

The PCB was designed for the container circuit to avoid messy wiring and ensure robustness of connections. The PCB aimed mostly at wiring and power distribution to the motor and switches from the battery, Arduino and L298 motor driver and hence, did not have any electronic components soldered on it. The board consisted of 2 layers of copper, 0.19 mm thick each, separated by a layer of non-conducting fiberglass. All the tracks were placed on the back layer to avoid the process of through-hole-plating to connect the two layers.

The fabrication of the PCB was done on the PCB CNC Mill with inputs as Gerber files (.grb) from the KiCad software used to design the PCB. Male header pins were selected as the connection method between the PCB and the electronic components. The standard footprint selected for the header pins was the Straight Pin Header with 0.1 inch (2.54 mm) Pitch. The pins were soldered on the front side after fabrication.

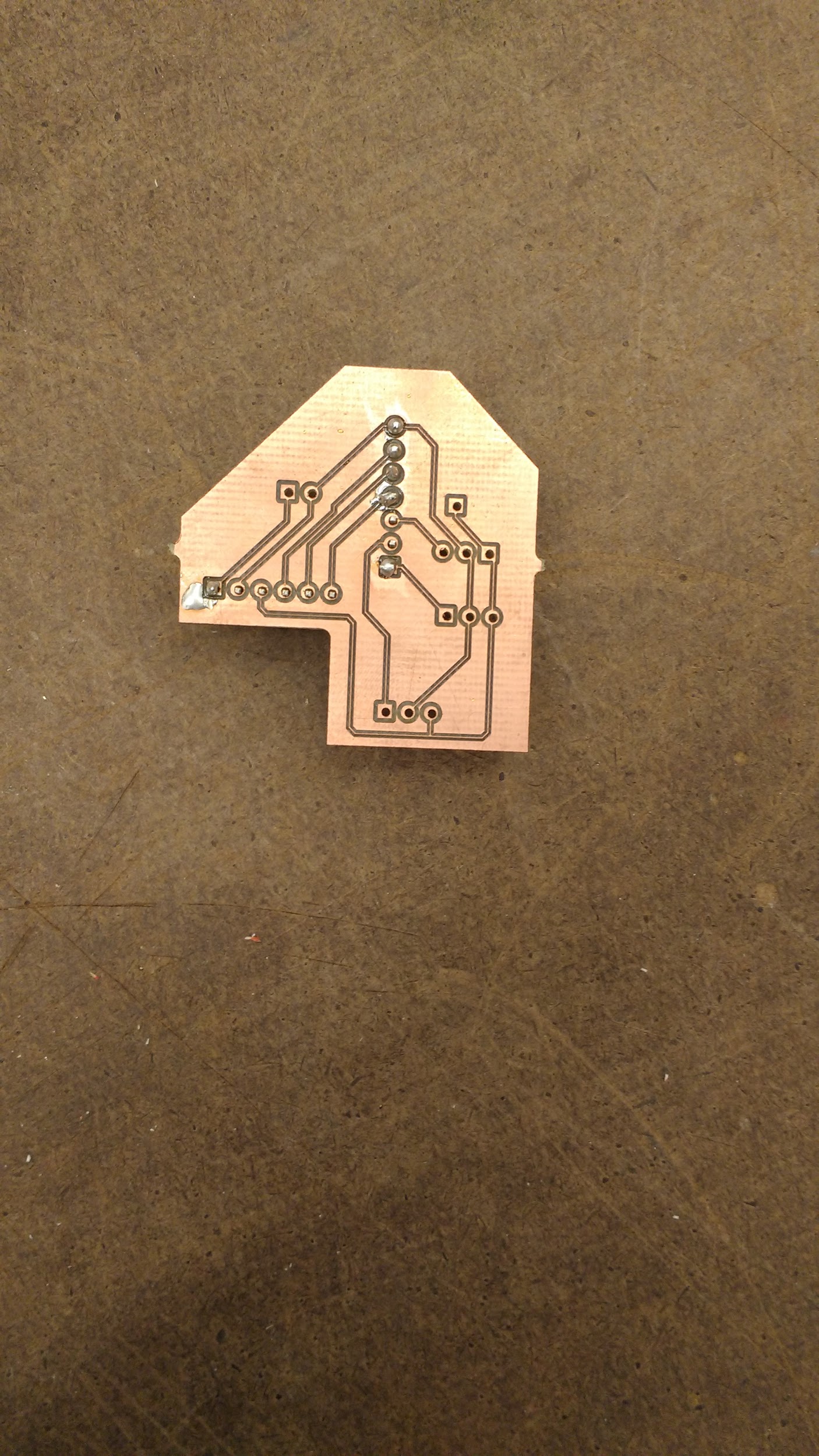
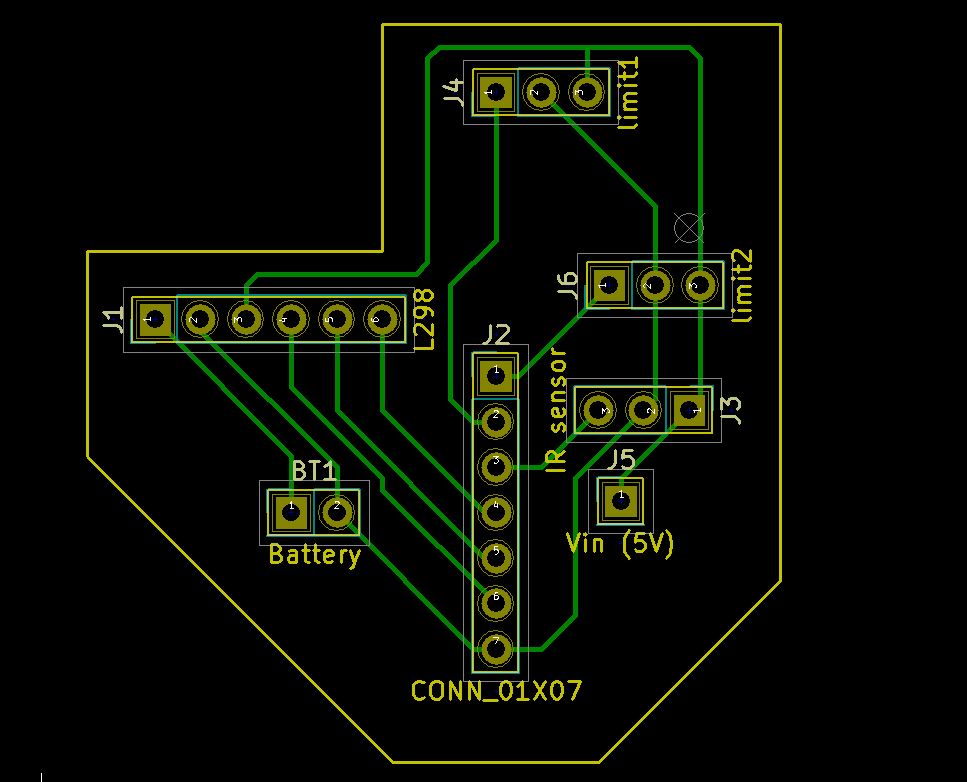


Figure 6 : PCB design

* 1. Actuators
     1. *Drive Motors*

For the drive system, two 12V DC motors with encoders were used. The motors had high rotational velocity and were powerful. Under no load, the speed was 251 rpm and torque was provided with use of a 43.7:1 metal gearbox. The motors were controlled with use of a L298N motor drive board.

* + 1. *Dispensing and Container Lead Screw Motor*

For the lead screw driving motor, a 12V DC worm geared motor was used. The motor was selected to have adequate torque to push heavy tennis balls. The limitation for the motor lead screw motor that was selected was that the slow rotational velocity. The slow velocity meant the push plate did not transfer enough energy to move the balls fast enough for sufficient momentum to roll into the container. As a result, the team created a software solution for dispensing the balls.

* + 1. *Trapdoor Motor*

A servo motor was used to rotate the trap door because it allow the team to easily control the specific location of the door. Moreover, the load of the door was minimal therefore a servo motor was the best option.

* 1. Sensors
     1. *Ball / Container Detection*

The ball detection mechanism was the core sensing part of the robot. Color-Sensitive Vision Sensor was used by utilizing a Pixy camera was used as the detection system. The image processing works by associating an object’s block size and coordinate within the sensors image to the location of the object. This yields the relative position of the object to the sensor. Preconditioning is required to make this association. Light conditions must be taken into consideration during the “training” of the camera. The Pixy was able to distinguish the difference between balls and the container by color differentiation. The camera was trained to associate the neon yellow color with tennis ball and the blue color with container. The limit of the image processing is the color associating. If the obstacles in the challenge round had been neon yellow or blue, the robot would have been confused and the task would not have been completed.

* + 1. *Lead screw switches*

To prevent the lead screw from moving beyond the limit and damage other components, two switches were added to regulate the movement range. Hinge lever roller, micro switches with rollers on the levers were used. The switches required a low push force and were robust for the application.

* + 1. *Wall Detection / Container Communication*

To prevent the robot from hitting the walls of the arena, the team introduced two IR sensor to the robot. In the searching mode, if either sensor detected something, changes could be made based on the signal.

To open and close the door of the container, the team used infrared signal as a remote control to operate the container door. Three infrared emitters were installed in the front and both sides of the robot. A receiver was placed at the center of the container door. In this way, the robot was able to communicate with the container to open and close the container door.

1. SOFTWARE

The main program of the robot implements a Finite State Machine (FSM) to simplify the programming logic. The five states in the FSM involved: BALL, COLLECT, CONTAINER, EMPTY and HALT. The program starts from BALL state, and after completing all the tasks the program ends in HALT state.

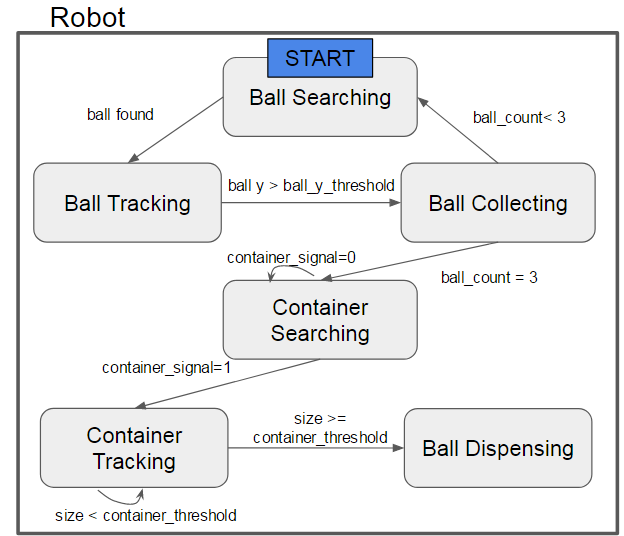


Figure 7 : Robot FSM

* 1. BALL state

In BALL state, the robot searched for the ball and move in the ball's direction once a target was identified. The search function relied on the data from the Pixy camera, which included the ball's “blob size” and center coordinates - x and y. If more than one ball was found, the robot moved to the closest one first. This was accomplished by selecting the target with the largest blob size - all balls have the same size and shape and if the ball appears bigger in the image than that ball is closer. The off-center error of the x coordinate of the target ball in the Pixy’s vision controlled the robot's steering by applying a proportional control of two motors' PWM differential with respect to this x-coordinate offset. If no ball was detected initially, the robot started rotating CW until a ball was found. As the robot approached the ball, the target ball's y coordinate increased gradually. Once that y coordinate met a dictated threshold, the program entered the COLLECT state, otherwise the logic remained in the BALL state.

* 1. COLLECT state

In the COLLECT state, the robot opened the front door and moved towards the ball to collect it. After successfully capturing the ball, the front door closed. This was done by monitoring the y coordinate of the ball. If it reached a certain threshold then disappeared, the ball was considered as captured. Then the state changed depend on the number of collected balls and the preset ball number. If all balls were collected, the program entered CONTAINER state. If there were more balls to collect, the program entered BALL state. If the robot lost track of the ball, the program entered BALL state as well.

* 1. CONTAINER state

In CONTAINER state, the robot worked the same as in BALL state, except that now it tracked the container rather than the ball. After the container's y-coordinate met the threshold, the robot stopped and the program entered EMPTY state.

* 1. EMPTY state

In EMPTY state, the robot first transmitted a series signal through the IR LEDs to open the container's gate. Then the robot’s front door was opened and the internal push plate was activated. Once all the balls were pushed out, the robot backed up a short distance, then closed the front door and accelerated forward to kick any balls remaining outside the container, into the container. Following that, another series signal was transmitted to close the container's gate. Finally, the program entered into the HALT state.

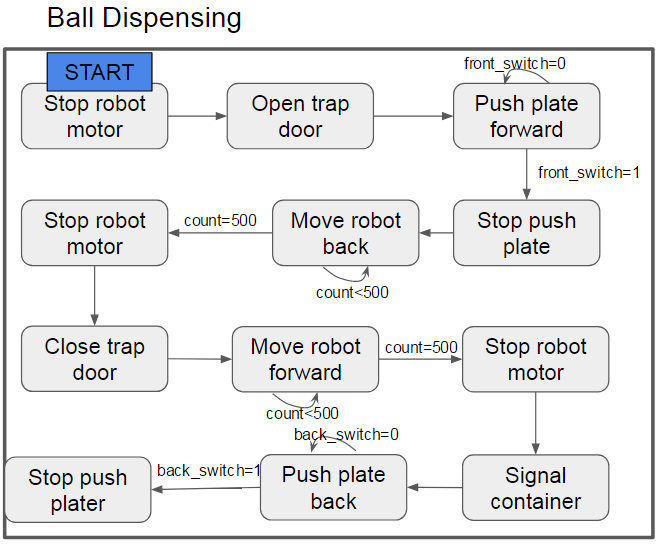


Figure 8 : Flowchart of the EMPTY state

* 1. HALT state

In HALT state, all motors were disabled to prevent the robot from moving. The robot was set to not read data from the Pixy cam. The thread remained indefinitely trapped in this state so that no unwanted behavior would occur.

1. RESULTS
   1. Experiments and Testing
      1. *Drive Motor Torque Testing*

Different types of motor were tested by manually applying friction to their shafts. The team’s initial intention was to use the motors with the worm gearbox to drive the robot. However, the team had several motors to choose from because of the opportunity to utilize previous ME588 components. All motors were tested with a 12 volts power supply. The motors with the worm gear box showed significant decrease in RPM with only slight external friction applied. The team determined that they were not suitable for driving the robot. A set of motors with 300 RPM rated at 12V showed almost no RPM drop with the same level of friction applied. These motors were selected to drive the robot. In the final competition, the motors performed well. The provided adequate power to even charge past obstacles in the challenge round of the competition. In the initial round of the competition there was some problem where the robot got stuck on the seam in the middle of the arena. This was an unlucky case. The left drive wheel was propped up and was not in contact with the ground. Therefore only the right drive wheel was engaged. Therefore propulsion of the robot to get out of the seam relied solely on one motor. One motor alone did not have the torque required to get the robot out of the stuck position.

* + 1. *Ball Detection Testing*

At the beginning, the team used servos motors for a pan and tilt mount for the Pixy to get more vision. First the center point of the pixy camera was used to subtract the x and y position of the object block in the Pixy vision to get the error, and use it to control the movement of the servo motors. After that, the angles the servos moved from center was used as the error. With this error, the differential speed of the robot drive motors was set so the object could be tracked. However, with the feedback loop, it increased the difficulties to control the robot. As a result, to robot could successfully track tennis balls, but failed to align with the balls perfectly. In other words, sometimes it might stop next to the ball, rather than in front of the ball, which made it unable to collect the ball. Therefore, the design was simplified and the pan and tilt mount was eliminated. This meant the broader range of vision was sacrificed. Without the servos for pan and tilt, the x axis error of the Pixy was directly used to build the differential speed for each drive motor. In this method, the robot rotated without forward action to search for the ball. Once aligned, it would go straight and collect it. The method worked very well. In the final competition many peers commented on how fast the robot was.

* + 1. *Ball Collection Performance Testing*

For the collection, the size of ball, detected in the Pixy vision, was initially size as the threshold to tell Arduino when to go into the COLLECT state. However, after several tests, the team found that the ball size was not a robust indicator since the size varied with the light conditions. The team modified the methods and used y-position of the object with the threshold. The serial monitor function was utilized to find out the desired value as the robot approached the tennis balls. After getting the desired value, it was applied in the program and tested. This method proved to be more robust for the COLLECTION state. The new method worked very well in the final competition. The robot did an excellent job of detecting and collecting balls in each round of the competition.

* + 1. *Ball Dispensing Performance Testing*

The initial concept for ball dispensing was to dispense the balls using just the push plate mechanism. However, the plate did not push the balls past the front plane of the robot. The team was concerned that if heavy balls were used, the push plate would not transfer enough energy to the balls for them to continue to roll into the container. In the prototype design, a plow was added to the plate that extended past the plane of the robot’s ball compartment. The design worked.

In final design, the plow was forgone and instead the problem was solved with software. The team found if the robot was moving fast enough, the balls would have enough momentum to continue rolling forward, out of the robot’s ball compartment and and into the storage container. The controls were set for the robot to stop a distance from the storage container, wait for the door to open, and then the robot zoomed forward stopping abruptly at the storage container door.

The push plate mechanism was still utilized in case the balls became trapped in the compartment. A final, third precaution was made to ensure the balls were dispensed and collected inside the storage container. The third measure was also through the controls. After the push plate was in the final front position, the robot was to back up and then ram forward to thrust any remaining balls into the storage container.

The three stage tactic to dispense the balls proved amusingly useful in the final victory round of the competition. Instead of detecting and avoiding the obstacles on the course, Team 8’s robot, with it’s strong motors, plowed across the obstacles. During the challenge round, one of the obstacles was actually collected inside the robot’s ball carrier compartment. The heavy obstacle was pushed on toward the container. When the initial push of the dispensing occurred, all of the trapped tennis balls rolled on into the container. Since the obstacle had no rolling momentum, it remained stuck inside the robot’s compartment. However, the second method of dispensing, the push plate, proved its usefulness. The obstacle was pushed to the front of the robot but still did not enter inside the storage container. No fret, the final safety net came into use. The robot, as designed, backed up and rammed forward pushing the obstacle inside the storage container right before the container closed shut. The final round truly tested the dispensing performance of the robot, the team was happy to see the the three stages of dispensing pay off for the hilarious ending.

* + 1. *Container Door Performance Testing*

The lead screw mechanism by itself worked well with the container door when tested without communication with the robot. The initial design was to make the robot approach the container in any orientation with respect to the door. Unfortunately, this was not possible due to more reasons than one. One such possibility was that the Pixy would identify the center of the door, but its orientation when dispensing the balls could make it collide with the lead screw set-up. Another situation could have been that the orientation of the robot could cause one of the balls to miss the container completely.

One solution was to align using Pixy with the help of angle detection; but Pixy gives erroneous and inaccurate results. After brainstorming, it was decided that the shape of the door should be changed. The new door was a cardboard sheet folded in the middle such that the folds were almost parallel to the container walls. Hence, when the robot approached the container from either side, the push-plate would be nearly parallel to the container walls which enabled the ball deposition to be almost always completely inside the container. This trick worked well.

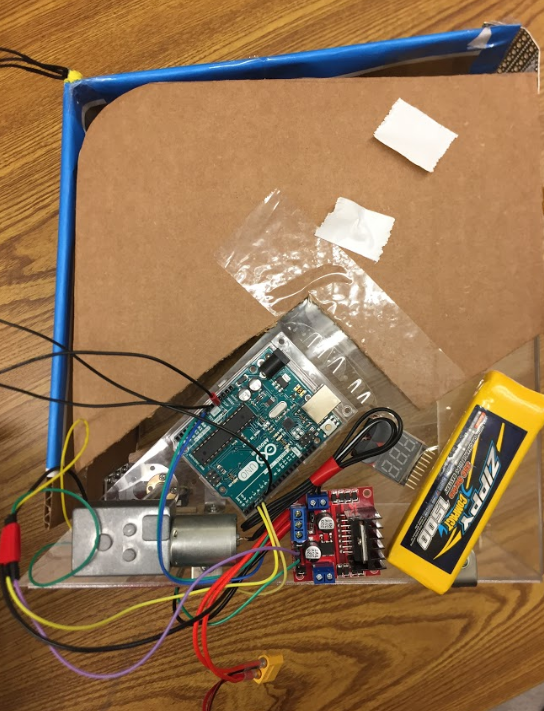
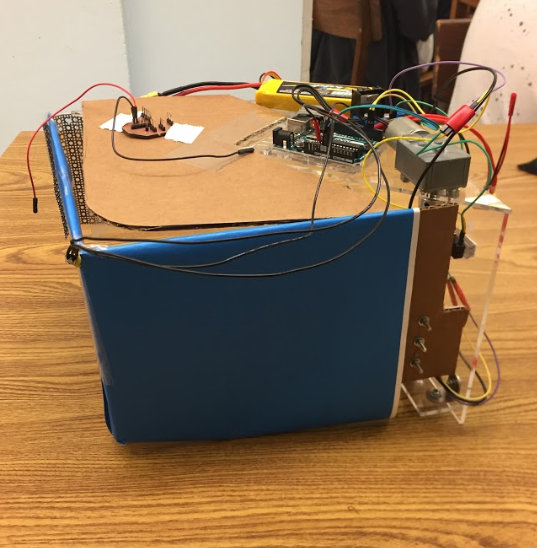


Figure 9 : Final Container Design

* 1. Budget

The team spent all of their Purdue budget plus some of their own monies for a total project cost of $340.00 in supplies. Most of the components were purchased through Amazon while a few were purchased through robot specific vendors. The team had many components on hand and utilized parts from previous ME588 projects. The a majority of the engineering time was spent in designing the robot and debugging the controls. Fabrication time and assembly time was minimal because of the use of laser cutting. The time spent in fabrication and assembly is not included in the budget. The full cost BOM can be found in the appendix.

* 1. Competition Performance

Team 8 came in second after the third round in the competition. In the first round, the heavy ball was collected with no issues. But as previously mentioned, the robot got stuck on the seam in the arena so the ball was not successfully dispensed. In the second round, both balls were collected however dispensing was not successful, the container door did not open. This was a similar case in the third round. The team learned that the room was too warm for the IR sensor on the container to function, the signal to open the container door was never recieved, the door did not open.

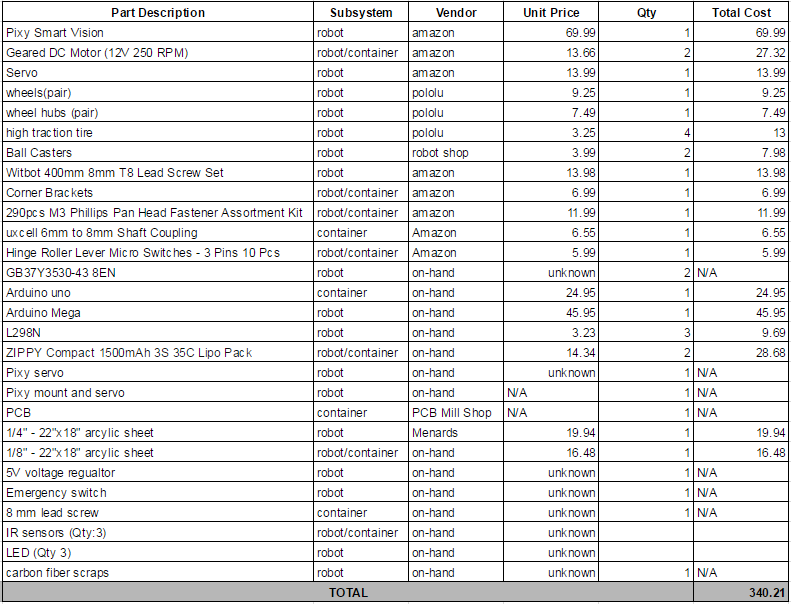
For the final challenge course, Team 8 modified the code to bypass the IR sensor and set it to open the container at the beginning of the round. While the team did not successfully avoid obstacles, as was previously discussed, the team was successful in retrieving two of the three balls and successfully dispensed the collected balls. Team 8 found more success than any other team in the challenge course and took the winner’s title.

1. LESSONS LEARNT AND CONCLUSION

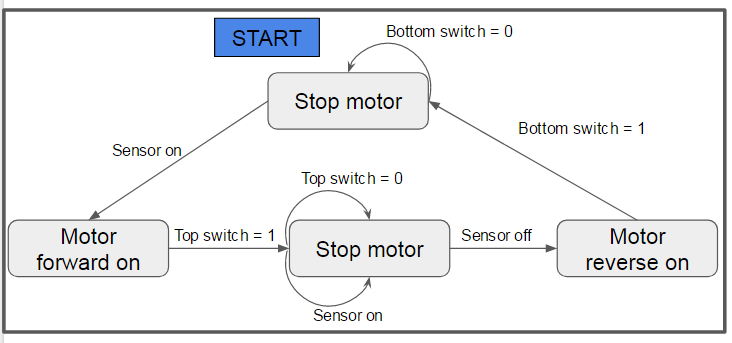
Team 8 learned a lot about mechatronic systems through the robot project. Image processing with the Pixy proved to be a good method for detecting balls but it was very sensitive to the surrounding lighting conditions. The Pixy alone is unreliable for alignment with a plane, this made it difficult for the team to get the robot to align with the container. In a future design additional sensors or switches would be added. The team found PCBs are a useful tool but in the design, the tracks should be spaced out to avoid shorting during soldering. The biggest learning the team had was in regards to the IR sensor and how it performed in the warm environment of the competition day. The large group of people caused the room to rise in temperature and IR sensor, used for the communication between the container and the robot, did not work. Future recommendations made by the team are to consider a more robust vision sensor and to consider a different method of remote communication between the container and the robot. Overall the team is satisfied with the design and were happy with their performance in the final competition.

1. APPENDIX

**Cost BOM**



**Container FSM**



**Robot Arduino Code**

#include <SPI.h>

#include <Pixy.h>

#include <Servo.h>

#include <IRremote.h>

#define DEBUG true

#define TIMEOUT 40\*1000

#define DELAY\_DOOR 30

#define SERVO\_CLOSE 100

#define SERVO\_OPEN 0

#define SIGNATURE\_BALL 1

#define SIGNATURE\_CONTAINER 2

#define DEFAULT\_BLOB\_SIZE 200

#define MIN\_BLOB\_SIZE 10

#define SPEED 140

//All pins

const int motorL1 = 7;

const int motorL2 = 8;

const int motorLPWM = 6;

const int motorR1 = 12;

const int motorR2 = 13;

const int motorRPWM = 11;

const int switchF = 3;

const int switchR = 4;

const int irFront = 0;

const int irRight = 7;

const int interruptPin = 2;

bool started;

int blob\_x;

int blob\_y;

int blobSize;

bool foundBlob;

int ballCount;

bool doorClosed;

int trackedBlock;

enum State {

BALL,

COLLECT,

CONTAINER,

EMPTY,

HALT

} state;

enum Dir {

FORWARD,

LEFT,

RIGHT,

BACKWARD

} dir;

Servo doorservo;

Pixy pixy;

IRsend irsend;

void setup()

{

if(DEBUG) {

Serial.begin(9600);

Serial.print("Initializing...\n");

}

pixy.init();

doorservo.attach(10);

pinMode(switchF, OUTPUT);

pinMode(switchR, OUTPUT);

pinMode(interruptPin, INPUT\_PULLUP);

attachInterrupt(digitalPinToInterrupt(interruptPin), eStop, FALLING);

foundBlob = false;

doorClosed = false;

ballCount = 0;

trackedBlock = -1;

state = BALL;

dir = FORWARD;

closeDoor();

pushIn();

stopMove();

stopPush();

if(DEBUG)

Serial.print("Ready\n");

}

void eStop() {

state = HALT;

}

void loop()

{

switch(state) {

case BALL:

findBall();

break;

case HALT:

stopMove();

break;

case COLLECT:

collectBall();

break;

case CONTAINER:

findContainer();

break;

case EMPTY:

empty();

break;

}

}

bool track(int signature) {

uint16\_t blocks;

blocks = pixy.getBlocks();

if (blocks){

trackedBlock = TrackBlock(blocks, signature);

if(trackedBlock == -1) {

return false;

}

blobSize = pixy.blocks[trackedBlock].width \* pixy.blocks[trackedBlock].height;

blob\_x = pixy.blocks[trackedBlock].x;

blob\_y = pixy.blocks[trackedBlock].y;

if(blobSize > MIN\_BLOB\_SIZE) {

return true;

}

}

return false;

}

int TrackBlock(int blockCount, int targetSignature) {

int trackedBlock = -1;

long maxblobSize = -1;

for (int i = 0; i < blockCount; i++) {

if (pixy.blocks[i].signature == targetSignature) {

long newblobSize = pixy.blocks[i].height \* pixy.blocks[i].width;

if (newblobSize > maxblobSize){

trackedBlock = i;

maxblobSize = newblobSize;

}

}

}

return trackedBlock;

}

void left(int PWM) {

int pin1 = LOW;

int pin2 = LOW;

if(PWM > 0) {

pin1 = HIGH;

} else if(PWM < 0){

PWM = -PWM;

pin2 = HIGH;

}

PWM = constrain(PWM, 0, 255);

digitalWrite(motorL1, pin1);

digitalWrite(motorL2, pin2);

analogWrite(motorLPWM, PWM);

}

void right(int PWM) {

int pin1 = LOW;

int pin2 = LOW;

if(PWM > 0) {

pin1 = HIGH;

} else if(PWM < 0){

PWM = -PWM;

pin2 = HIGH;

}

PWM = constrain(PWM, 0, 255);

digitalWrite(motorR1, pin1);

digitalWrite(motorR2, pin2);

analogWrite(motorRPWM, PWM);

}

void moveMotor(int leftPWM, int rightPWM) {

left(leftPWM);

right(rightPWM);

}

void stopMove() {

moveMotor(0, 0);

}

void breakMove() {

digitalWrite(motorL1, LOW);

digitalWrite(motorL2, LOW);

analogWrite(motorLPWM, 255);

digitalWrite(motorR1, LOW);

digitalWrite(motorR2, LOW);

analogWrite(motorRPWM, 255);

delay(200);

stopMove();

}

void FollowBlock(int trackedBlock) {

int forwardSpeed = 140;

// Steering differential is proportional to the off center value

int differential = 160 - pixy.blocks[trackedBlock].x;

// Adjust the left and right speeds by the steering differential.

int leftSpeed = constrain(forwardSpeed - 0.8\*differential, -SPEED, SPEED);

int rightSpeed = constrain(forwardSpeed + 0.8\*differential, -SPEED, SPEED);

moveMotor(leftSpeed, rightSpeed);

}

void ScanForBlocks() {

static uint32\_t lastMove = 0L;

static uint32\_t lastStraight = 0L;

static int forwardCount = 0;

if (millis() - lastMove > 20) {

int frontDist = analogRead(irFront);

int rightDist = analogRead(irRight);

lastMove = millis();

if(rightDist > 500) {

dir = LEFT;

} else {

if(dir == LEFT) {

dir = FORWARD;

} else if(millis() - lastStraight > 2000){

{

if(forwardCount < 10) {

forwardCount++;

dir = FORWARD;

}

else {

dir = RIGHT;

forwardCount = 0;

lastStraight = millis();

}

}

} else{

dir = RIGHT;

}

}

if(frontDist > 500) {

dir = BACKWARD;

forwardCount = 0;

lastStraight = millis();

}

switch(dir) {

case FORWARD:

moveMotor(160, 160);

break;

case LEFT:

moveMotor(-100, 100);

break;

case RIGHT:

moveMotor(100, -100);

break;

case BACKWARD:

moveMotor(-200, -200);

break;

}

}

}

void findBall() {

bool found = track(SIGNATURE\_BALL);

if(found) {

if(!foundBlob)

breakMove();

foundBlob = true;

FollowBlock(trackedBlock);

if( blob\_y > 100 ) {

state = COLLECT;

if(DEBUG)

Serial.println("BALL >>> COLLECT");

}

} else if(!foundBlob){

//search for the ball

if(DEBUG)

Serial.println("Searching balls");

ScanForBlocks();

}

}

void collectBall() {

bool found = track(SIGNATURE\_BALL);

if(found) {

FollowBlock(trackedBlock);

openDoor();

} else if(blob\_y > 190){

delay(400);

if(closeDoor()) {

ballCount++;

foundBlob = false;

}

if(ballCount >= 3) {

state = CONTAINER;

if(DEBUG) {

Serial.println("COLLECT >>> CONTAINER");

}

} else {

state = BALL;

if(DEBUG) {

Serial.println("COLLECT >>> BALL");

}

}

}

}

bool openDoor() {

if(doorClosed) {

doorservo.write(SERVO\_OPEN); //change number here

if(DEBUG)

Serial.println("Opened door.");

doorClosed = false;

return true;

}

return false;

}

bool closeDoor() {

if(!doorClosed) {

doorservo.write(SERVO\_CLOSE); //change number here

if(DEBUG)

Serial.println("Closed door.");

doorClosed = true;

return true;

}

return false;

}

void findContainer() {

bool found = track(SIGNATURE\_CONTAINER);

if(found){

if(!foundBlob) {

breakMove();

openContainer();

openDoor();

}

foundBlob = true;

FollowBlock(trackedBlock);

Serial.println(blob\_y);

if( blob\_y > 100 ) {

state = EMPTY;

if(DEBUG)

Serial.println("CONTAINER->EMPTY");

}

} else if(!foundBlob) {

ScanForBlocks();

}

}

void openContainer() {

Serial.println("Opened container.");

for (int i = 0; i < 50; i++) {

irsend.sendSony(0xaaa,12);

delay(40);

}

}

void closeContainer() {

Serial.println("Closed container.");

for (int i = 0; i < 5; i++) {

irsend.sendSony(0xbbb,12);

delay(40);

}

}

int getFrontDistance() {

return analogRead(irFront);

}

int getRightDistance() {

return analogRead(irRight);

}

void empty() {

breakMove();

openContainer();

openDoor();

pushOut();

moveMotor(-140, -140);

delay(600);

breakMove();

closeDoor();

moveMotor(200, 200);

delay(600);

stopMove();

moveMotor(-140, -140);

delay(600);

breakMove();

delay(1000);

closeContainer();

pushIn();

state = HALT;

}

void pushOut() {

digitalWrite(switchF, HIGH);

while(digitalRead(switchF));

stopPush();

}

void pushIn() {

digitalWrite(switchR, HIGH);

while(digitalRead(switchR));

stopPush();

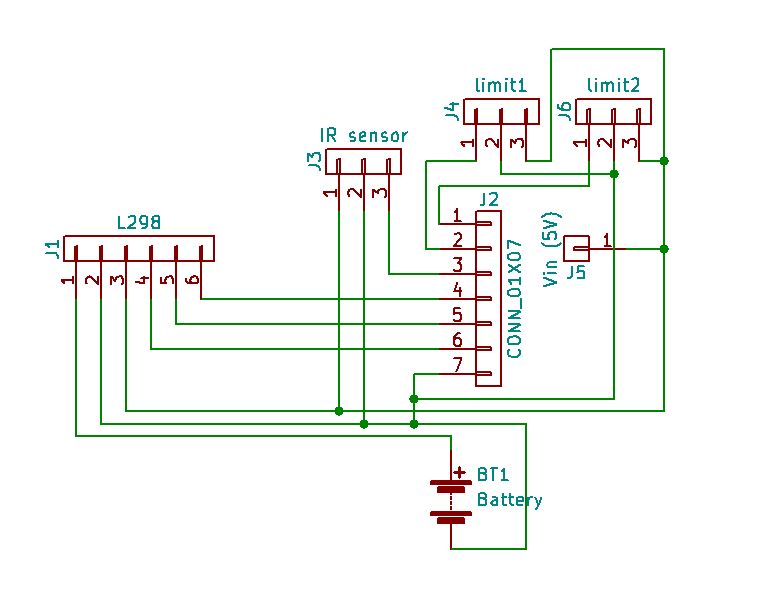
}

void stopPush() {

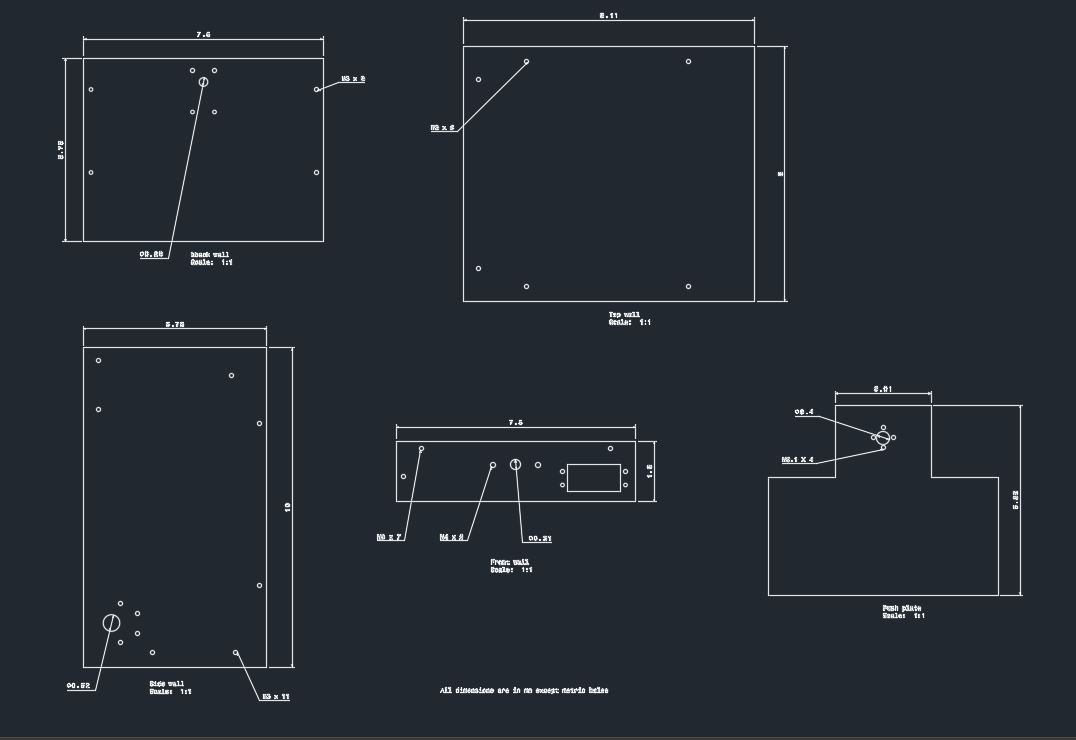
digitalWrite(switchF, LOW);

digitalWrite(switchR, LOW);

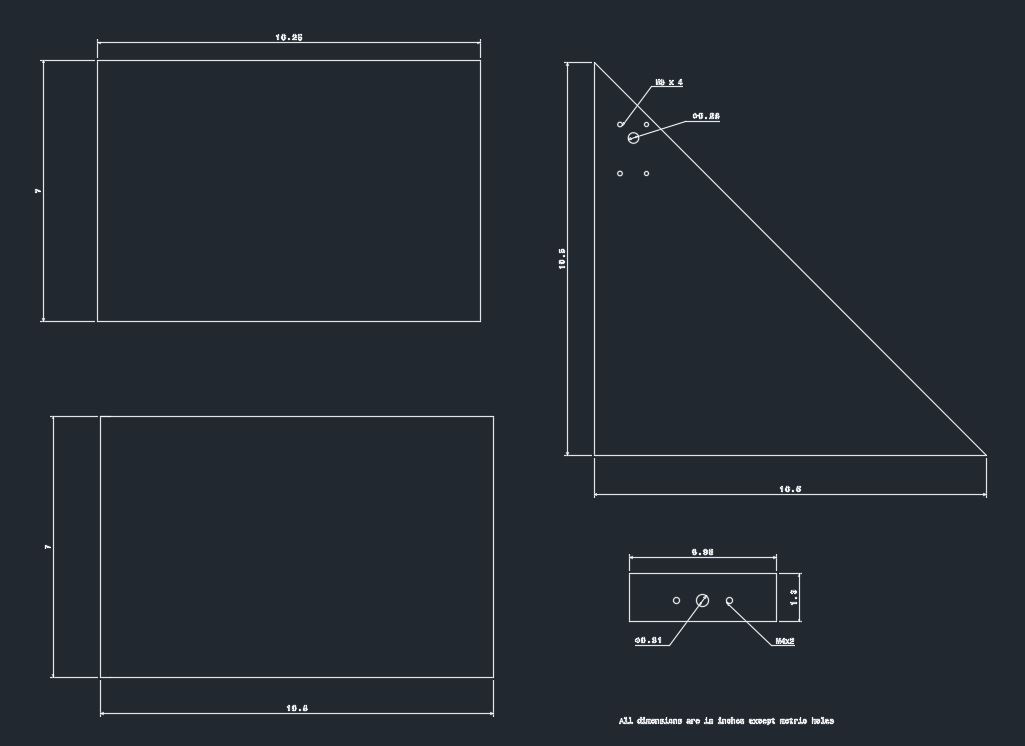
}



Container circuit diagram for PCB



AUTOCAD file for robot chassis



AUTOCAD file for container