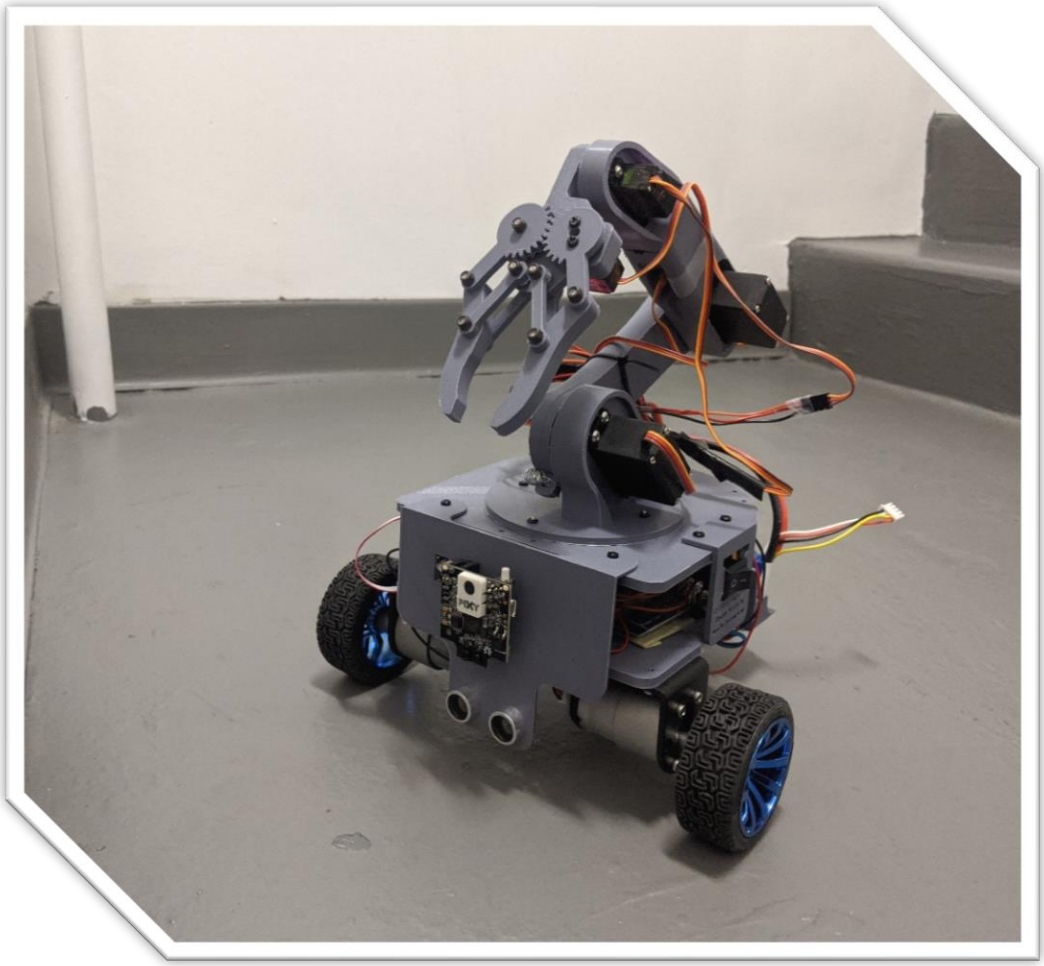
# Mobile Robotic Arm

Gene Nadela

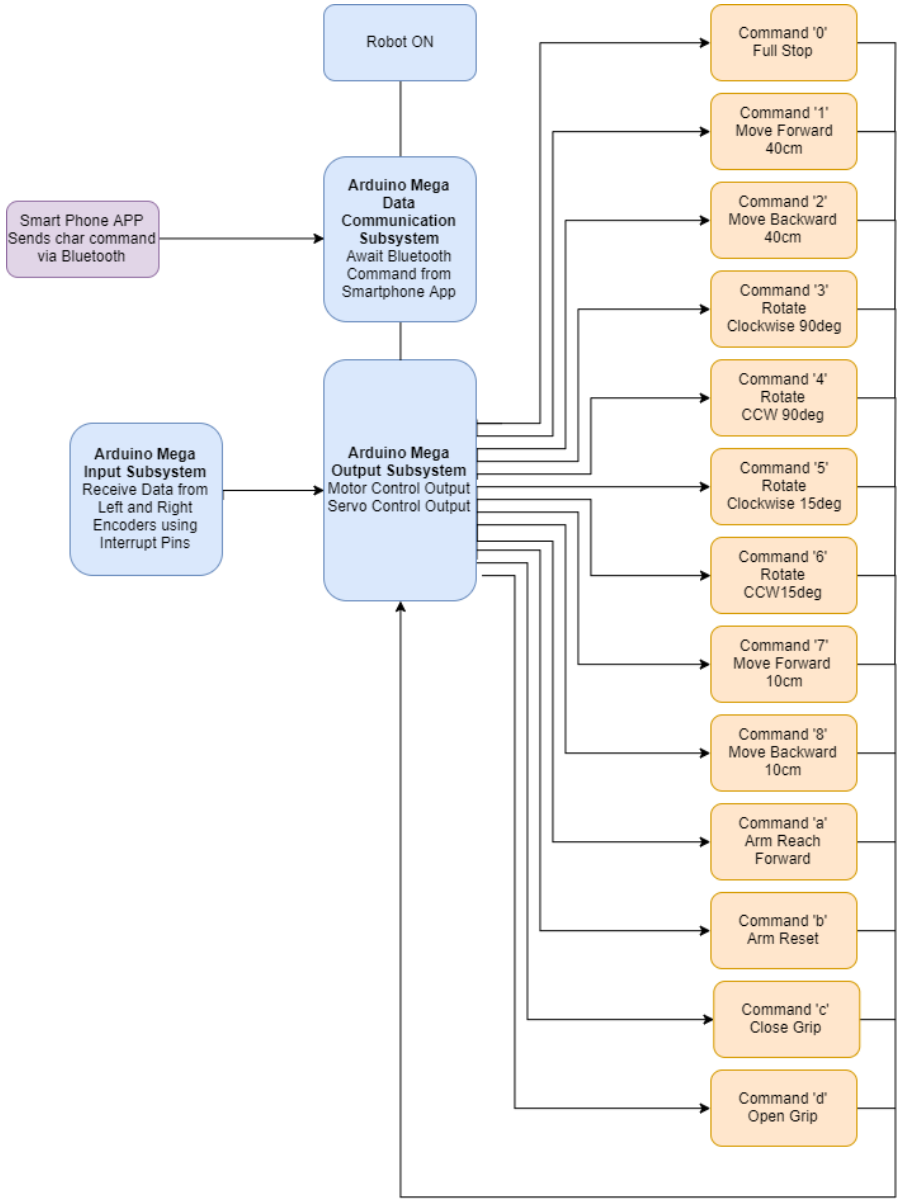


CET4811 – E388 – Capstone Design Project

Fall 2019 – Dr. Farrukh Zia

# Table of Contents

# Project Summary




### Operation Description

The mobile arm robot follows commands as issued by a smartphone application via Bluetooth. A number of preset commands have been applied to the program to allow the user varying degrees of control regarding position of the robot, placement of the arm, and engagement of the robot's end-effector/gripper. When the robot is finished executing a single command, it will await further instructions.

### Potential Applications

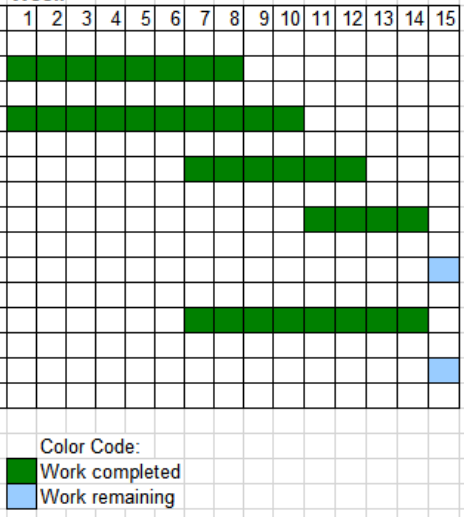
A mobile robot with multiple points of articulation may be beneficial in industries where the movement of hazardous goods or material that may harm a human worker, may be operated remotely. The robot may also see some use in aiding the physically impaired, allowing the user to acquire objects they may not be able to physically grasp on their own using a touchscreen interface for control.

# Project Management Plan

Project Name: Autonomous 4 DOF Sorting and Transport Robot

Group Member Names: Gene Nadela

| Task / Sub task | Person Responsible | Week |
|---|---|---|
| 1: Mechanical Design | Gene Nadela | |
| 2. Electrical Design (Sensor & Motor Interface) | Gene Nadela | |
| 3. Software Design (Control Program) | Gene Nadela | |
| 4. Testing and Debugging | Gene Nadela | |
| 5. Project Demonstration | Gene Nadela | |
| 6. Project Report | Gene Nadela | |
| 7. Project Presentation | Gene Nadela | |




The project is being produced by a single person, as such, all responsibilities will belong to that person.

Initial mechanical design will undergo multiple iterations and redesigns over the course of 5 to 6 weeks, while waiting electrical components to be sourced from international shipping. As components arrive, such as motors and encoders, tests were to be performed on each to verify nominal operation of the component as well as further research gathered for software implementation in the final few weeks of the project,

# Mechanical Design

## Concept Drawings

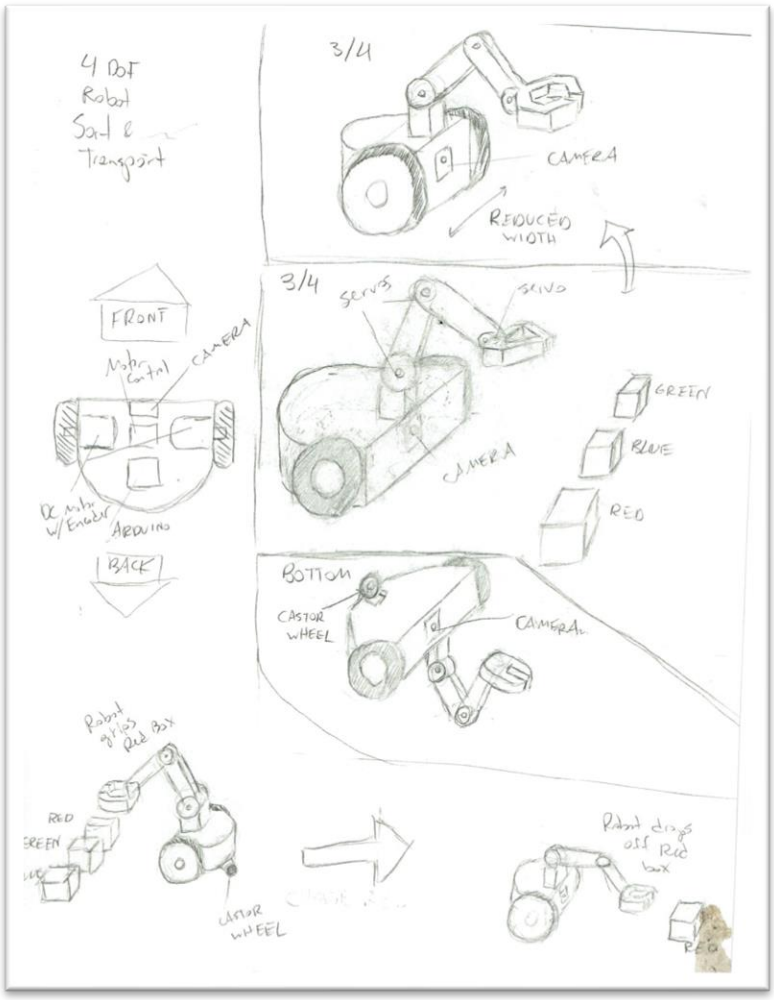



*Fig. 1 - Initial concept drawings for robot. The main objective of robot was to create a mobile platform for a functional robot with multiple degrees of freedom.*

## Construction

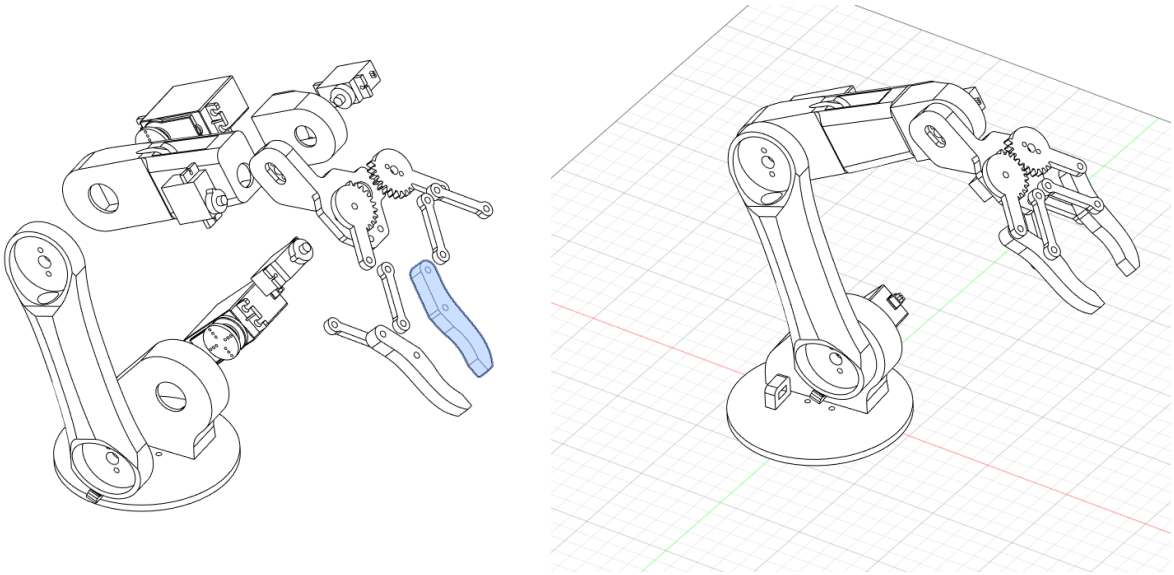


*Fig 2. - Robotic arm is mechanically the most complicated portion of the system, and most prone to mechanical failure. Plastic parts after 3D print, are examined thoroughly for defects, and trimmed with tools such as sanders and blades so that they move fluidly with one another.*

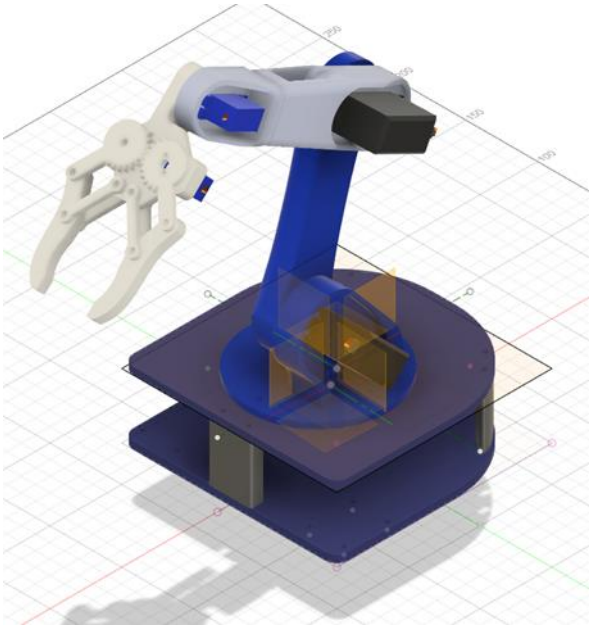


*Fig. 3 - The assembled robotic arm is fit onto the upper platform of the mobile platform.*

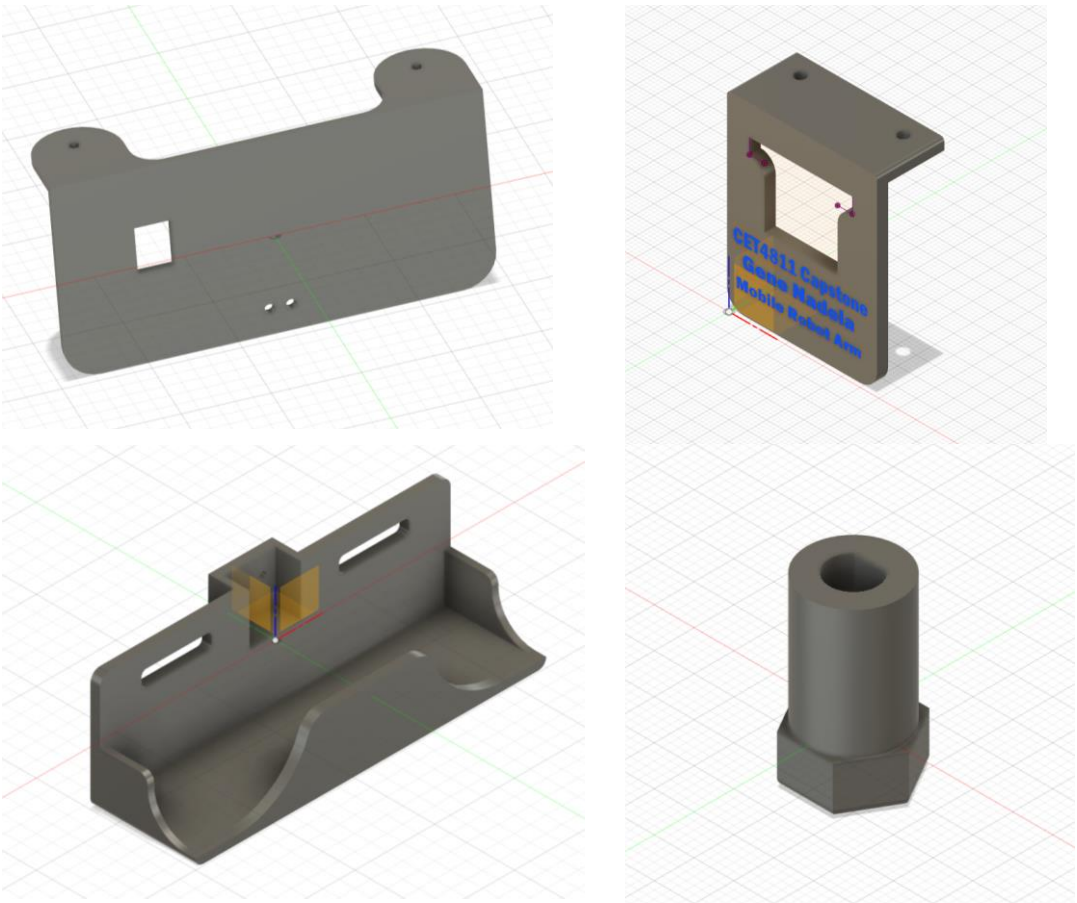



*Fig. 4 - Additional parts were developed as the need arose, in order to better fit specific components to the over construction. The front cover (top left) is build to conceal and protect any wiring protruding from the front of the robot. The name plate (top right) also houses the main power switch. The battery holder (bottom left) houses the battery, and the wheel coupler (bottom right) allows the wheels to be bound to the D shaft of the motor.*

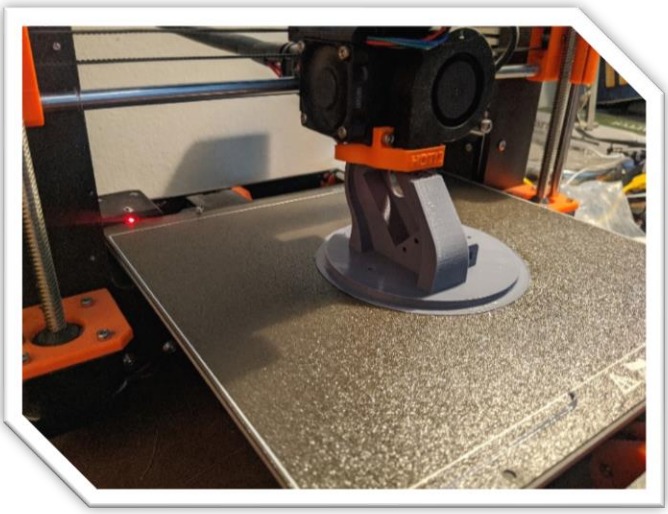


*Fig. 5a - Shoulder mount for upper platform being printed on Prusa Mk3 3D home printer*.

*Fig. 5b - Planning initial mounting options for robotic platform, while additional parts continued to be printed and tested.*

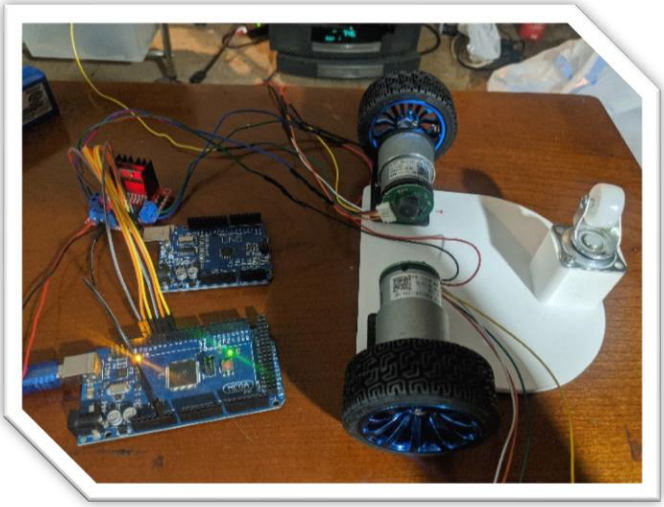


*Fig. 5c - The DC motors and encoders were mounted on the bottom component platform and tested using the motor driver and Arduino Mega.*

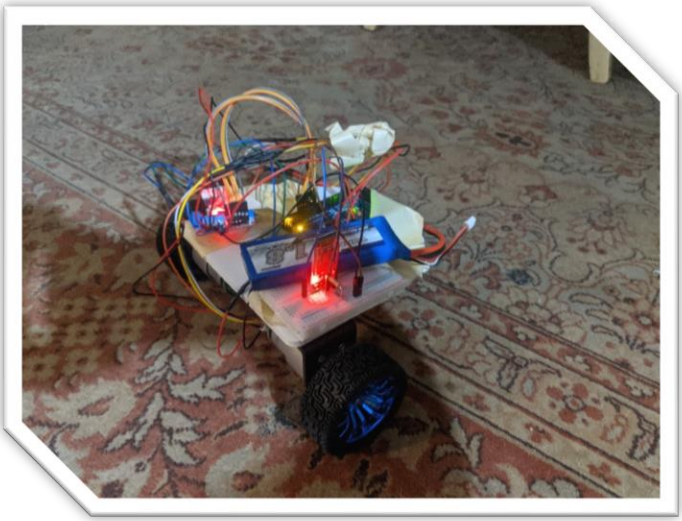


*Fig. 5d - With the Arduino, Motor Driver, Bluetooth Module and Battery mounted on component platform, remote control drive is tested on the platform to verify control and speed. This initial testing phase confirms the use of higher voltage battery due to the need for increased torque as a result of the increased weight of the platform.*

*Fig. 5e - Platform was redesigned for additional components and to support more weight, as well as to aesthetically hide wiring.*

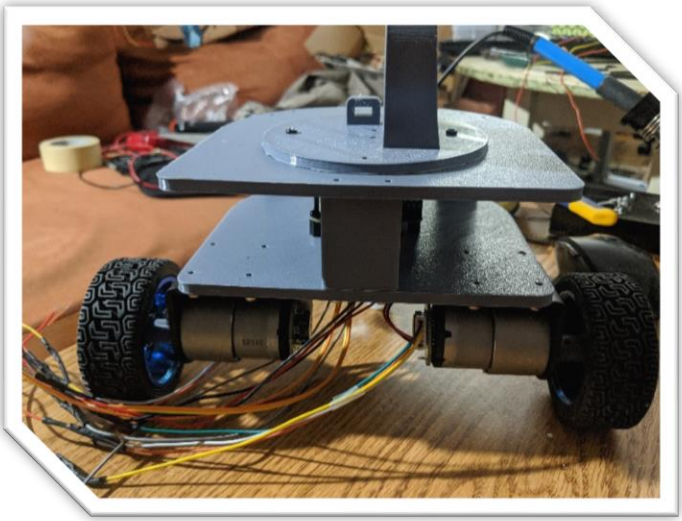


*Fig. 5f - The motor driver is attached to the bottom of the bottom platform, where additional wiring may be hidden. This also served to lower the robots CoM (Center of Mass) to better promote balance.*

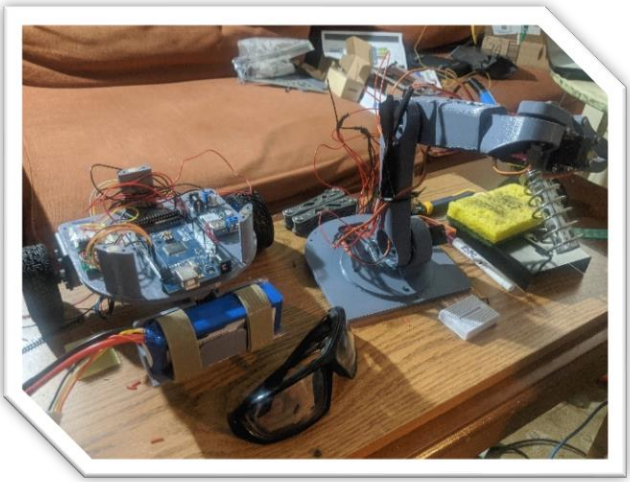


*Fig. 5g - A battery holder was designed to shift weight further from the front of the robot, as mounting the arm resulted in significant balancing issues when the arm is fully extended.*

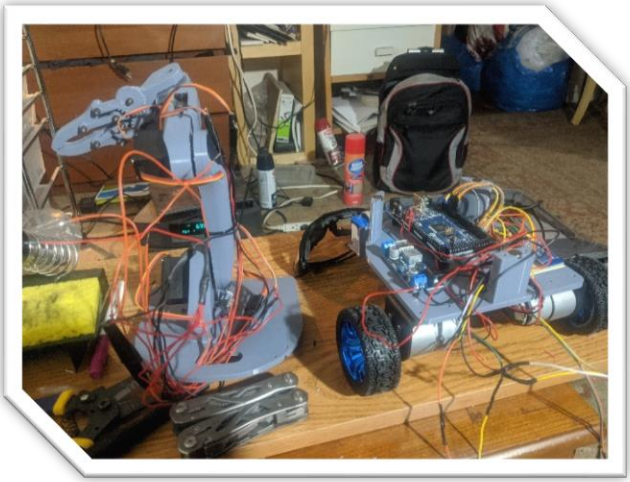


*Fig. 5h - To facilitate quicker troubleshooting, the mounting points on the bottom plat and top plate are fitted with high strength N52 Neodymium Magnets to keep the upper platform in place, while allowing for easier access to the robot's internal electrical structure.*

*Fig. 5i - Nearly complete construction, ready for additional testing for software development and mechanical improvements.*

**Design Measurements:**

Arm Length (Full Extension): 40 cm
Width(Left Wheel to Right Wheel): 25 cm
Length (Front to Back of Battery): 20 cm
Height (Bottom of Wheel to Top of Shoulder): 19 cm
Wheel Diameter: 68mm
Shaft Diameter: 5mm D Shaft
Weight: 1.18 kg

Parts List

| Components | Source |
|---|---|
| 5DOF Robotic Arm | STL Design files from HowToMechatronics.com<br>Printed using PLA+ on Prusa Mk3 Home 3D Printer |
| Lower Component Platform | Designed on Autodesk Fusion 360 CAD Software.<br>Printed using PLA+ on Prusa Mk3 Home 3D Printer. |
| Upper Arm Shoulder Platform | Designed on Autodesk Fusion 360 CAD Software.<br>Printed using PLA+ on Prusa Mk3 Home 3D Printer. |
| Battery Holder | Designed on Autodesk Fusion 360 CAD Software.<br>Printed using PLA+ on Prusa Mk3 Home 3D Printer. |
| Front Cover Protector | Designed on Autodesk Fusion 360 CAD Software.<br>Printed using PLA+ on Prusa Mk3 Home 3D Printer. |
| 4mm D Shaft Coupler | Designed on Autodesk Fusion 360 CAD Software.<br>Printed using PLA+ on Prusa Mk3 Home 3D Printer. |
| Castor Wheel | Ace Hardware Store |
| M3 Screws, Standoffs | Amazon.com Store – Sutemribor |
| N52 Neodymium Magnets | Amazon.com Store - ELECFIND |

| **Troubleshooting** **Problem**: Wrist Lift not reliably lifting gripper mechanism. |
|---|
| Examination of the physical connection between servo and servo horn shows mechanical wear of the internal gearing. The servo was replaced to remedy the problem. |
| **Troubleshooting** **Problem**: Fully extended arm causes system to lose balance. |
| In order to maintain balance, attention must be given to the systems CoM, and the areas where the object is heaviest. Among the heaviest components in the system is the battery and the Arduino Mega microcontroller. These objects were situated towards the back of the system, and a battery holder was designed to extend the weight further away from the CoM in order to counterbalance the weight of the robot arm when fully extended. |

# Electrical Design

## Embedded Controller

- ELEGOO MEGA 2560 R3 Board ATmega2560 – Arduino Mega Microcontroller board, has 54 digital I/O pins (with 14 pins that can be used as PWM outputs). 16 analog inputs, 4 UART, 16 MHz crystal oscillator, ICSP header. Accepts 7 to 12 volt input, operating at 5 volts. Each pin has a 40 mA limit on current. 128kB Flash Memory, 8KB SRAM, 4KB EEPROM.

## Input Sensors

- Hall-effect Motor Encoders – each encoder has two Hall-effect sensors 90 degrees out of phase, meaning they may be used to determine the direction and velocity of the motor simultaneously. The sensor accepts 5V supply voltage, and has a single data line from each hall sensor marked Sensor A and Sensor B.

## Output Actuators and Devices

- DC Motor 37-520 – 12V DC motor with a rated power of 4.8W. Has a rated current of 360mA with a stall current of 2.8A. It has a rated torque of 1.0 kgfcm. It is a reduction assembly, meaning there is an internal DC motor at a speed of 11000RPM, and is reduced at a 1:30 ratio.
- MG995 Servo Motor – Servo motor with an operational voltage of 4.8V to 7.2V. Has a stall torque of 8.5 kgfcm to 10kgfcm. Uses PWM control signals to change the position of the servo motor.
- MG90 Servo Motor - Servo motor with an operational voltage of 4.8V to 7.2V. Has a stall torque of 1.8 kgfcm to 2.2kgfcm. Uses PWM control signals to change the position of the servo motor.

## Input and Output interface circuits

- Motor Driver 2A Dual L298 H-Bridge – Operates in 5V logic, and drives motors from 5V to 35V, with a maximum single bridge current of 2A. Uses an L298N Double H bridge chip, which allows it to drive two separate DC motors at either forward or backwards operation, which is determined by their associated enable pins. The enable pins also allow for speed control using PWM signals.

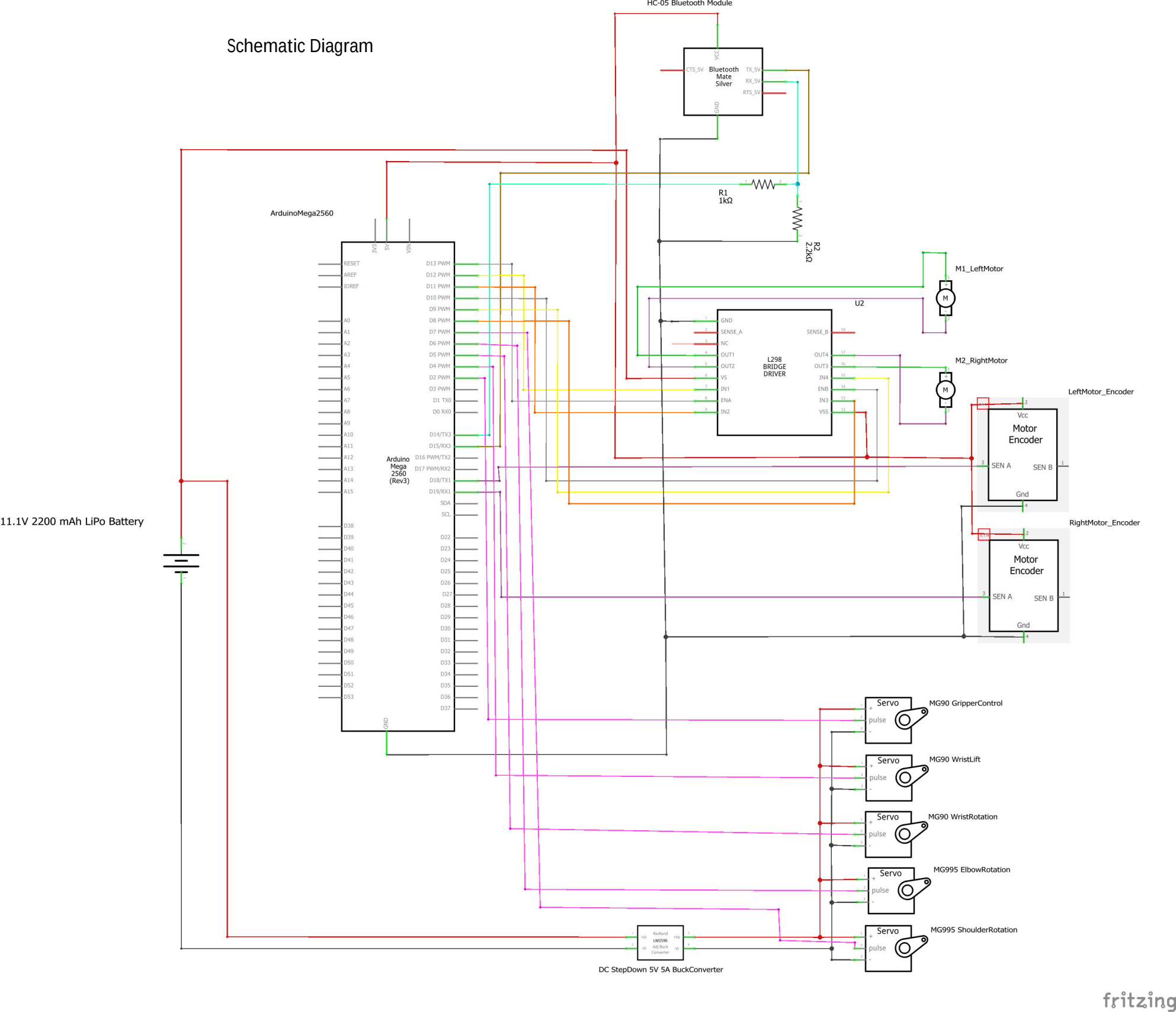
## Data communication and network interface

- HC-05 Wireless Bluetooth Host Serial Transceiver Module – operates at a supply voltage of 3.3V to 6.0V, and a working current of 30mA. Has a operating range of 10 meters. Adds full duplex, two way wireless functionality to system. When powered on, the module is discoverable by any Bluetooth device.

## Power system

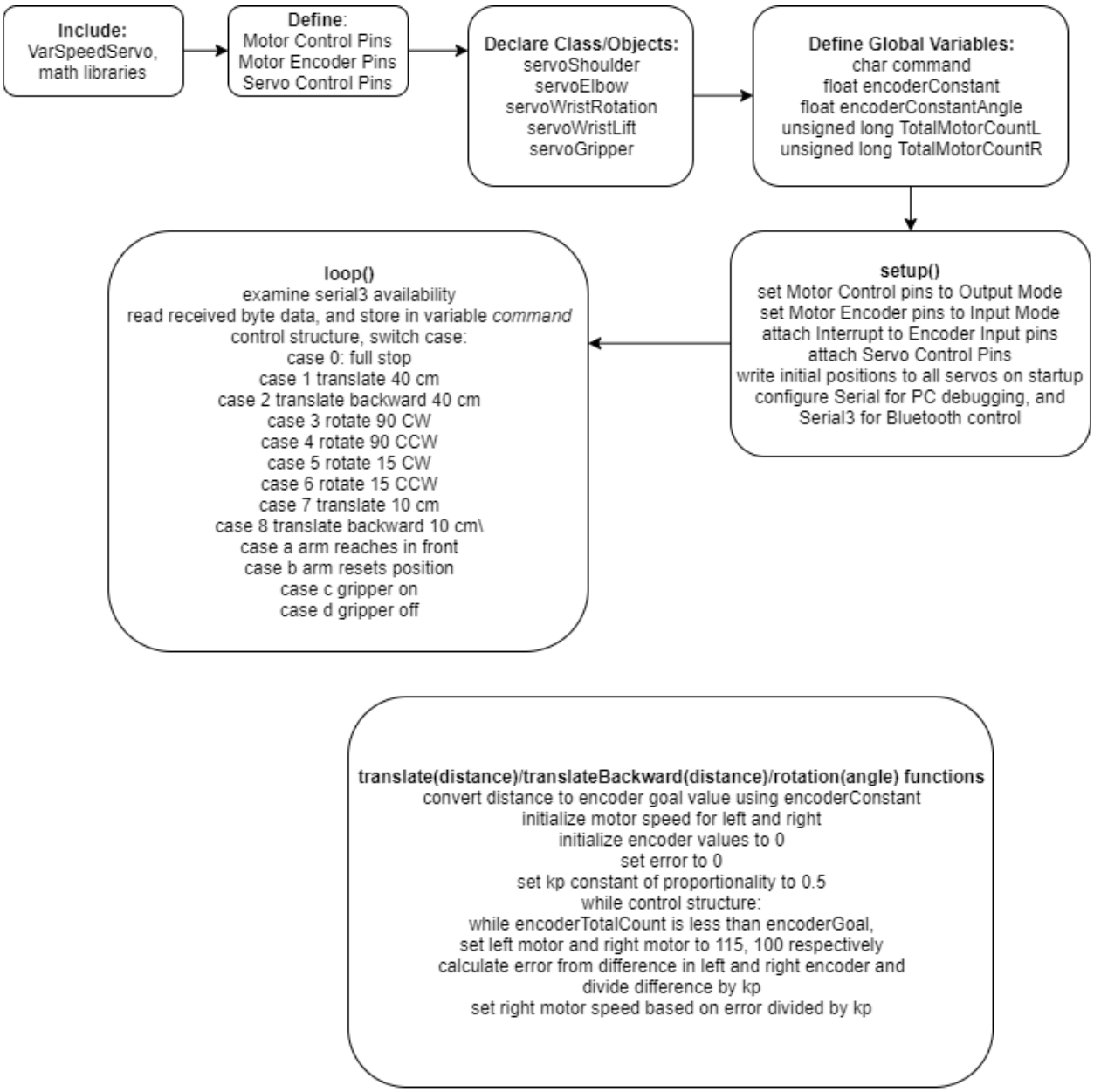
- DROK USB Buck Converter Voltage Regulator DC 9V-36V Step Down to DC 5V – accepts an input voltage of 8V to 36V, and steps down the voltage efficiently to 5.2V.
- Turnigy High Discharge LiPo Battery 3Cell/11.1V, 2200 mAh – Supplies 11.1V of voltage at a discharge rating between 25C to 35C.

# Schematic Diagram



# Software Design

## Mobile Robot Arduino Code – Diagram




## Mobile Robot Arduino Code - Source

```
/***
 * Autonomous 4DOF Robot
 * Code Description: Control code for Arduino-based mobile robot with arm.
 * Hardware Required: 2-DC Motors with Encoders, L298N Motor Driver, 2 MG995, 3 MG90
 * IDE version: 1.8.10
 * Programmer name: Gene Nadela
```

```
 * November 3, 2019
 * v. 1.0.0
 ***/

// pre-processor directives
#include <VarSpeedServo.h>
#include <math.h>

// pin name definitions

//Motor Control Pins
const int enaPinL = 13; //PWM enable pin for left motor, Grey Wire, ENA-A
const int in1PinL = 12;
const int in2PinL = 11; //Orange Wire
const int enaPinR = 10; //PWM enable pin for right motor, Grey Wire, ENA-B
const int in1PinR = 9;
const int in2PinR = 8; //Orange Wire

//Motor Encoder Pins
const int encAL = 18; //Encoder A, Left Motor
//const int encBL = 19; //Encoder B, Left Motor
const int encAR = 19; //Encoder A, Right Motor
//const int encBR = 21; //Encoder B, Left Motor

//Servo Control Pins
const int servoShoulderPin = 7;
const int servoElbowPin = 6;
const int servoWristRotatePin = 5;
const int servoWristLiftPin = 4;
const int servoGripperPin = 2;

// class/object declarations
VarSpeedServo servoShoulder;
VarSpeedServo servoElbow;
VarSpeedServo servoWristRotation;
VarSpeedServo servoWristLift;
VarSpeedServo servoGripper;

// function declarations
void setMotorLeft(int speed, boolean reverse);
void setMotorRight(int speed, boolean reverse);
void fullStop();
void moveForward();
void moveBackward();
void rotateCW();
void rotateCCW();
void translate(float dist);

// global variables
char command; //Receive character commands from remote control system
const float encoderConstant = 3000;
const float encoderConstantAngle = 150;
volatile unsigned long TotalMotorCountL = 0; //initialize left motor encoder count
volatile unsigned long TotalMotorCountR = 0; //initialize right motor encoder count

void setup() {
  // configure hardware peripherals
  //Motor Controls
  pinMode(enaPinL, OUTPUT);
  pinMode(in1PinL, OUTPUT);
  pinMode(in2PinL, OUTPUT);
  pinMode(enaPinR, OUTPUT);
  pinMode(in1PinR, OUTPUT);
```

```
  pinMode(in2PinR, OUTPUT);

  //MotorEncoder
  pinMode(encAL, INPUT);
  //pinMode(encBL, INPUT);
  pinMode(encAR, INPUT);
  //pinMode(encBR, INPUT);

  //MotorEncoder Interrupt Pins
  attachInterrupt(digitalPinToInterrupt(encAL), countLeftMotor, FALLING);
  attachInterrupt(digitalPinToInterrupt(encAR), countRightMotor, FALLING);

  //Servo Motor Control
  servoShoulder.attach(servoShoulderPin);
  servoElbow.attach(servoElbowPin);
  servoWristRotation.attach(servoWristRotatePin);
  servoWristLift.attach(servoWristLiftPin);
  servoGripper.attach(servoGripperPin);

  //Servo Motor Initialized Position
  servoShoulder.write(140, 10, false);
  servoElbow.write(90, 10, false);
  servoWristRotation.write(110, 10, false);
  servoWristLift.write(0, 10, false);
  servoGripper.write(45, 10, false);

  // configure data communication
  Serial.begin(115200); //PC Serial Communication for Debugging
  Serial3.begin(9600); //Bluetooth Serial Communication for Remote Control
}

void loop() {

    if (Serial3.available() > 0) {
      command = Serial3.read();
        switch (command) {

        case '0':
          fullStop();// stop all motors
          Serial.print("Stop");
        break;

        case  '1':
          translate(0.4); // move forwards 40 cm
          Serial.print("Forward 40cm");
        break;

        case '2':
          translateBackwards(0.4); // move backwards 40 cm
          Serial.print("Backward 40cm");
        break;

        case '3':
          rotate(1.57); //CW 90 degrees
          Serial.print("CW 90deg");
        break;

        case '4':
          rotate(-1.57); //CCW 90 degrees
          Serial.print("CCW 90deg");
        break;

        case '5':
```

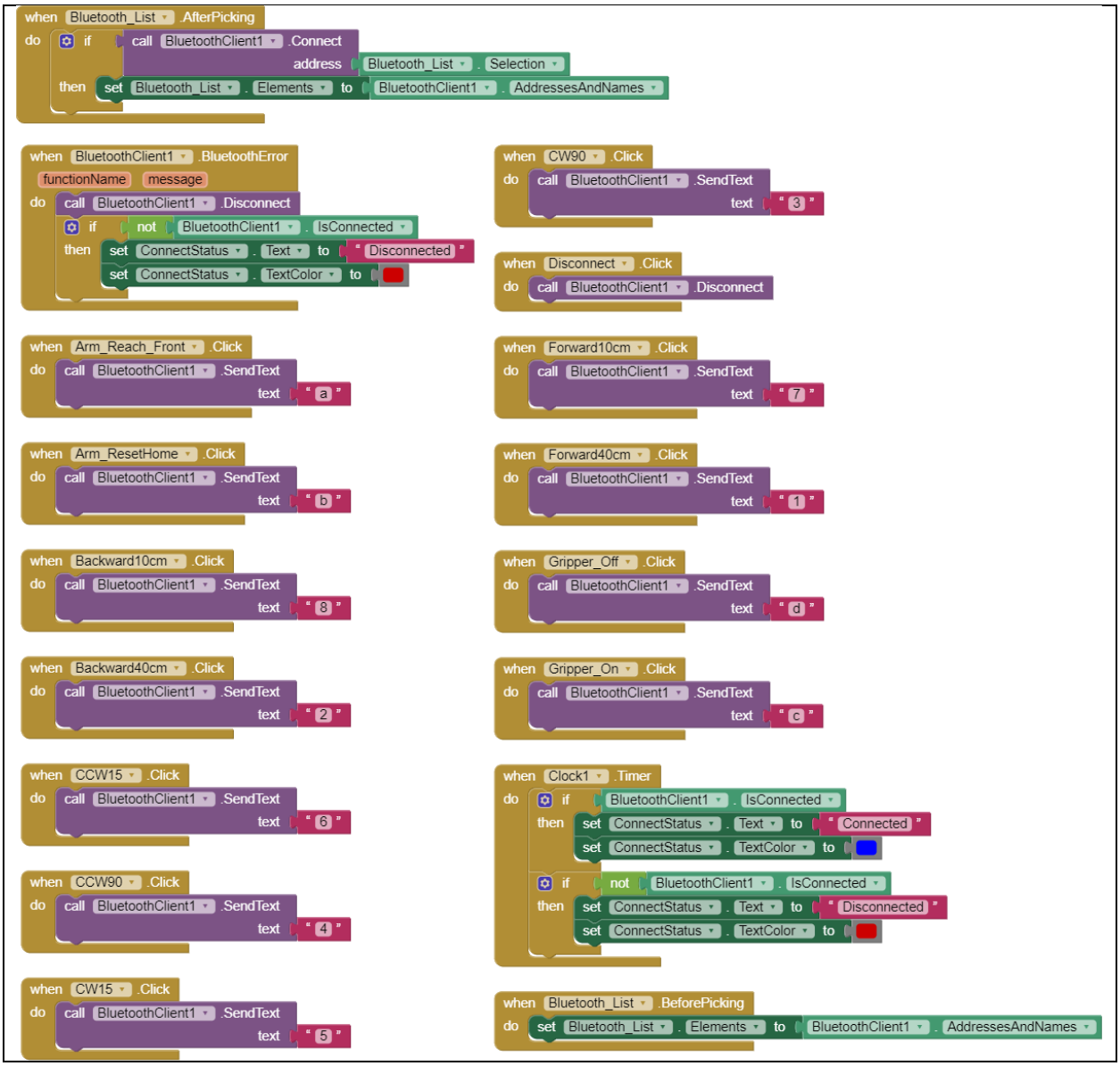
```
          rotate(0.26); //CW 15 degrees
          Serial.print("CW 15deg");
        break;

        case '6':
          rotate(-0.26); //CCW 15 degrees
          Serial.print("CCW 15deg");
        break;

        case '7':
          translate(0.1); //Forward 10 cm
          Serial.print("Forward 10cm");
        break;

        case '8':
          translateBackwards(0.1); //Backward 10 cm
          Serial.print("Backward 10cm");
        break;

        case 'a':
          armReachFront();// arm reach towards front
        break;

        case 'b':
          armReset();// set arm to home position
        break;

        case 'c':
          gripOn();// close grip
        break;

        case 'd':
          gripOff();// open grip
        break;

        default:
        // do nothing for other command values
        break;
        }
    }
}

//Function Definitions

void armReachFront() {
  servoShoulder.write(40, 20, false);
  servoElbow.write(110, 20, false);
  servoWristRotation.write(110, 20, false);
  servoWristLift.write(90, 20, false);
}

void armReset() {
  servoShoulder.write(140, 20, false);
  servoElbow.write(90, 20, false);
  servoWristRotation.write(110, 20, false);
  servoWristLift.write(0, 20, false);
}

void gripOn() {
  servoGripper.write(0, 20, false);
}

void gripOff() {
```

```
  servoGripper.write(45, 20, false);
}

void setMotorLeft(int speed, boolean reverse){ //sets velocity of left motor
  analogWrite(enaPinL, speed);
  digitalWrite(in1PinL, !reverse);
  digitalWrite(in2PinL, reverse);
}
void setMotorRight(int speed, boolean reverse){ //sets velocity of right motor
  analogWrite(enaPinR, speed);
  digitalWrite(in1PinR, !reverse);
  digitalWrite(in2PinR, reverse);
}
void moveForward() { //Move Forward
  setMotorLeft(115, false);
  setMotorRight(115, false);
}
void moveBackward() { //Move Backward
  setMotorLeft(100, true);
  setMotorRight(100, true);
}
void rotateCW() { //Rotate Clockwise
  setMotorLeft(127, true);
  setMotorRight(127, false);
}
void rotateCCW() { //Rotate CounterClockwise
  setMotorLeft(127, false);
  setMotorRight(127, true);
}
void fullStop() { //Total Stop Robot
  setMotorLeft(0, false);
  setMotorRight(0, false);
}
void countLeftMotor() {
  TotalMotorCountL++;
}
void countRightMotor() {
  TotalMotorCountR++;
}


void translate(float dist) { //moves robot forward in meters
  float encoderGoal = dist*encoderConstant;
  int encoderTotalCount = 0;
  int speedMotorLeft = 115; //initialize left motor speed
  int speedMotorRight = 100; //initialize right motor speed
  TotalMotorCountL = 0; //initialize encoder value left
  TotalMotorCountR = 0; //initialize encoder value right
  int error = 0; //initialize error
  float kp = 0.5; //constant of proportionality
  while(encoderTotalCount < encoderGoal) {
    setMotorLeft(speedMotorLeft, false);
    setMotorRight(speedMotorRight, false);
    error = TotalMotorCountL - TotalMotorCountR;
    speedMotorRight += error/kp;
    TotalMotorCountL = 0; //initialize encoder value left
    TotalMotorCountR = 0; //initialize encoder value right
    Serial.print("TotalMotorL: ");
    Serial.print(TotalMotorCountL);
    Serial.print("   TotalMotorR: ");
    Serial.print(TotalMotorCountR);
    Serial.print("   LeftSpeed: ");
    Serial.print(speedMotorLeft);
```

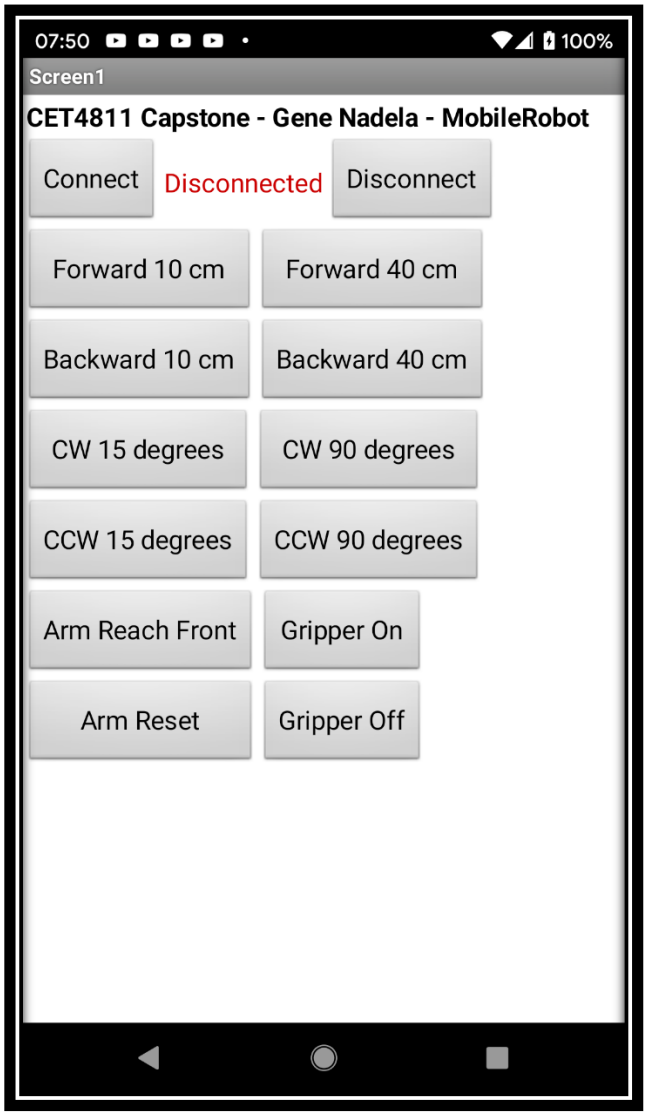
```
    Serial.print("   RightSpeed: ");
    Serial.print(speedMotorRight);
    Serial.print("   Error: ");
    Serial.println(error);
    delay(10);
    encoderTotalCount += TotalMotorCountL;
  }
  fullStop();
}

void translateBackwards(float dist) { //moves robot backwards in meters
  float encoderGoal = dist*encoderConstant;
  int encoderTotalCount = 0;
  int speedMotorLeft = 115; //initialize left motor speed
  int speedMotorRight = 100; //initialize right motor speed
  TotalMotorCountL = 0; //initialize encoder value left
  TotalMotorCountR = 0; //initialize encoder value right
  int error = 0; //initialize error
  float kp = 0.5; //constant of proportionality
  while(encoderTotalCount < encoderGoal) {
    setMotorLeft(speedMotorLeft, true);
    setMotorRight(speedMotorRight, true);
    error = TotalMotorCountL - TotalMotorCountR;
    speedMotorRight += error/kp;
    TotalMotorCountL = 0; //initialize encoder value left
    TotalMotorCountR = 0; //initialize encoder value right
    Serial.print("TotalMotorL: ");
    Serial.print(TotalMotorCountL);
    Serial.print("   TotalMotorR: ");
    Serial.print(TotalMotorCountR);
    Serial.print("   LeftSpeed: ");
    Serial.print(speedMotorLeft);
    Serial.print("   RightSpeed: ");
    Serial.print(speedMotorRight);
    Serial.print("   Error: ");
    Serial.println(error);
    delay(10);
    encoderTotalCount += TotalMotorCountL;
  }
  fullStop();
}

void rotate(float angle) { //rotate robot in radians
  float encoderGoal = fabs(angle*encoderConstantAngle);
  int encoderTotalCount = 0;
  int speedMotorLeft = 115; //initialize left motor speed
  int speedMotorRight = 100; //initialize right motor speed
  TotalMotorCountL = 0; //initialize encoder value left
  TotalMotorCountR = 0; //initialize encoder value right
  int error = 0; //initialize error
  float kp = 0.5; //constant of proportionality

  if (angle >= 0) {
    while(encoderTotalCount < encoderGoal) {
    setMotorLeft(speedMotorLeft, false);
    setMotorRight(speedMotorRight, true);
    error = TotalMotorCountL - TotalMotorCountR;
    speedMotorRight += error/kp;
    TotalMotorCountL = 0; //initialize encoder value left
    TotalMotorCountR = 0; //initialize encoder value right
    Serial.print("TotalMotorL: ");
    Serial.print(TotalMotorCountL);
    Serial.print("   TotalMotorR: ");
```

```
    Serial.print(TotalMotorCountR);
    Serial.print("   LeftSpeed: ");
    Serial.print(speedMotorLeft);
    Serial.print("   RightSpeed: ");
    Serial.print(speedMotorRight);
    Serial.print("   Error: ");
    Serial.println(error);
    delay(10);
    encoderTotalCount += TotalMotorCountL;
    }
  fullStop();
  }
  else {
    while(encoderTotalCount < encoderGoal) {
    setMotorLeft(speedMotorLeft, true);
    setMotorRight(speedMotorRight, false);
    error = TotalMotorCountL - TotalMotorCountR;
    speedMotorRight += error/kp;
    TotalMotorCountL = 0; //initialize encoder value left
    TotalMotorCountR = 0; //initialize encoder value right
    Serial.print("TotalMotorL: ");
    Serial.print(TotalMotorCountL);
    Serial.print("   TotalMotorR: ");
    Serial.print(TotalMotorCountR);
    Serial.print("   LeftSpeed: ");
    Serial.print(speedMotorLeft);
    Serial.print("   RightSpeed: ");
    Serial.print(speedMotorRight);
    Serial.print("   Error: ");
    Serial.println(error);
    delay(10);
    encoderTotalCount += TotalMotorCountL;
    }
  fullStop();
  }

}
```

Smartphone App – MIT App Inventor



Smartphone App UI




| **Troubleshooting** **Problem**: Robot not driving straight. |
| --- |
| Initial thought was one of the motors was slower than the other due to manufacturing defects. Attempted to compensate by increasing the default forward speed. Results continued to be inconsistent regardless of further adjustments to either motor. Further research into the subject yielded the realization that the problem is common, particularly in robots using differential drive. Implemented software solution using constant of proportionality (kp) error correction, effectively slaving the right motor with the left motor by comparing encoder data, and correcting speed in real time. |

# Appendix

## Electrical Components

- **ELEGOO MEGA 2560 R3 Board ATmega2560**
  https://www.robotshop.com/media/files/pdf/arduinomega2560datasheet.pdf
- **Hall-effect Motor Encoders**
  https://www.banggood.com/37-520-High-Torque-Reducer-AB-Dual-Phase-Hall-Encoder-DC-Motor-for-Smart-Car-DIY-Part-p-1390247.html
- **DC Motor 37-520**
  https://www.banggood.com/37-520-High-Torque-Reducer-AB-Dual-Phase-Hall-Encoder-DC-Motor-for-Smart-Car-DIY-Part-p-1390247.html
- **MG995 Servo Motor**
  https://www.electronicoscaldas.com/datasheet/MG995_Tower-Pro.pdf
- **MG90 Servo Motor**
  https://engineering.tamu.edu/media/4247823/ds-servo-mg90s.pdf
- **Motor Driver 2A Dual L298 H-Bridge**
  https://www.instructables.com/id/Arduino-Modules-L298N-Dual-H-Bridge-Motor-Controll/
- **HC-05 Wireless Bluetooth Host Serial Transceiver Module**
  http://www.electronicaestudio.com/docs/istd016A.pdf
- **DROK USB Buck Converter Voltage Regulator DC 9V-36V Step Down to DC 5V**
  https://nettigo.eu/products/dc-dc-step-down-converter-module-cx8571-9-36v-to-5-2v-6a-usb-fast-charge

# References


Websites

http://andrewjkramer.net/motor-encoders-arduino/

https://blog.robotiq.com/kinematics-why-robots-move-like-they-do

https://howtomechatronics.com/tutorials/arduino/diy-arduino-robot-arm-with-smartphone-control/

https://www.robotc.net/wikiarchive/Tutorials/Arduino_Projects/Mobile_Robotics/VEX/Using_encoders_to_drive_straight


Components Purchased at:

| |
|---|
| Ace Hardware Store – Mechanical Components |
| Tinkersphere – Electrical Components |
| Amazon.com – Mechanical/Electrical Components |