Midterm Presentation

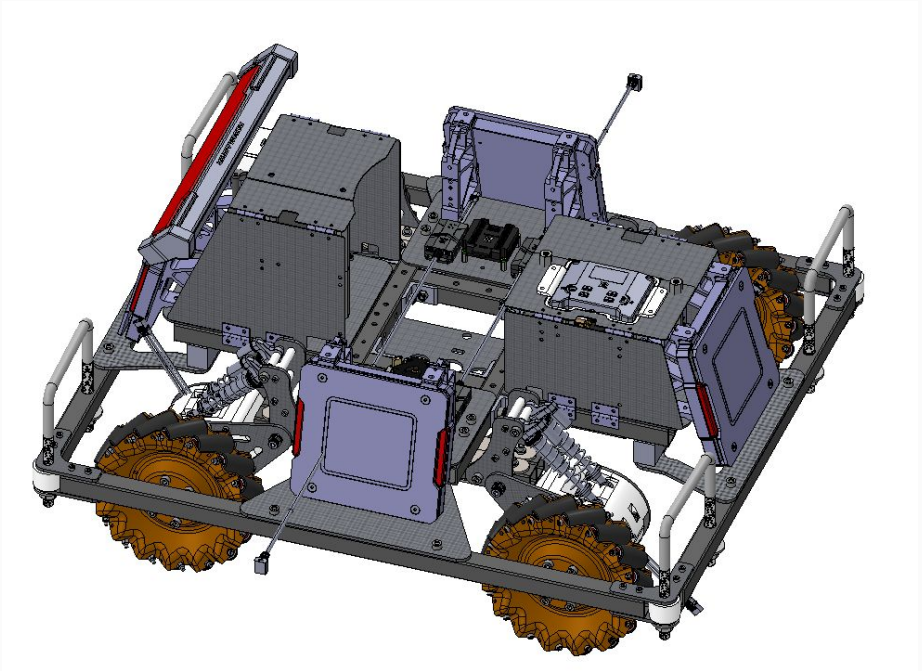
Triton Robotics
University of California, San Diego

Table of Contents

Infantry

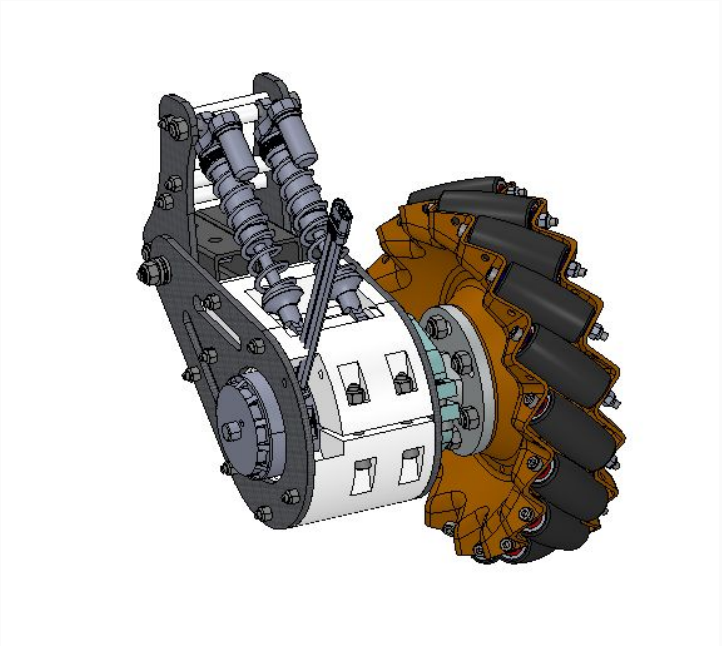
CAD Design - Chassis

- **Chassis**
  - **Capability:**
    - **Protection for and Easy Access of Electronics**
    - **Relatively stable under violent crash**
    - **Roller on corners of chassis to prevent getting stuck in arena corners**
    - **U-bolts for Rescue Mechanism**
  - **To be improved:**
    - **Wire Management (tried using cable clips to ameliorate)**
    - **Reduce Size**



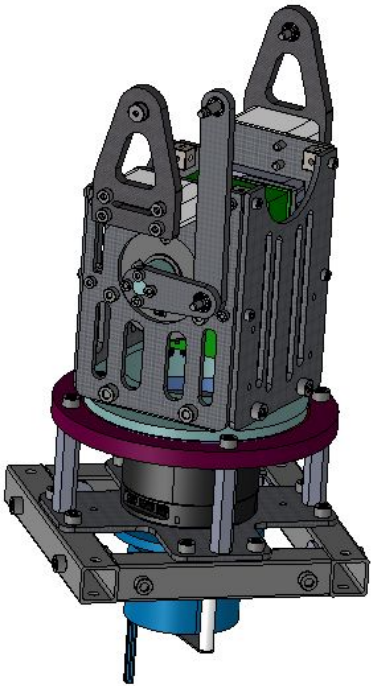
CAD Design - Suspension

- **Suspension**
  - **Capability:**
    - **Reduces Vibration during Movement (Especially in Front and Back Translational Movement)**
    - **Keeps Mecanum Wheels on Ground**
    - **Ease of Mounting to Chassis**
  - **To be improved:**
    - **Modularity (tried using press-fit nuts to reduce quantity of fasteners)**



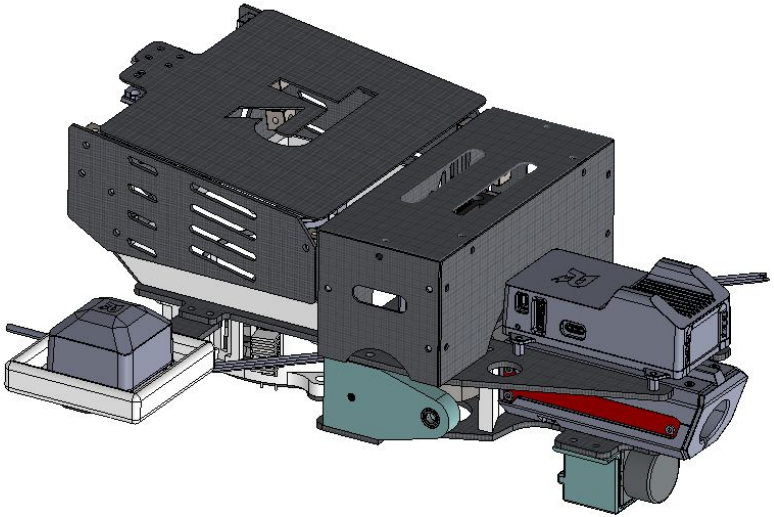
CAD Design - Turret Base

- **Turret Base**
  - **Capability:**
    - **Allows Rotation of Chassis Relative to Turret with Slip Ring**
    - **Ease of Access of Electronics**
    - **Reduced Moment of Inertia due to New Pitch Motor Position**
  - **To be improved:**
    - **Reduce number of fasteners (trying to include tapped holes in machined parts)**



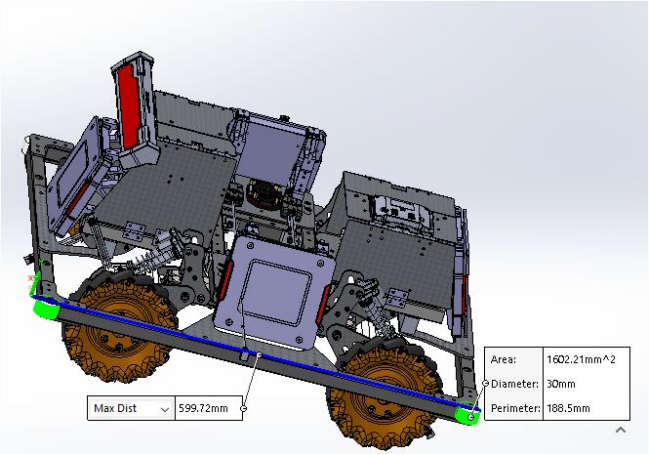
CAD Design - Turret

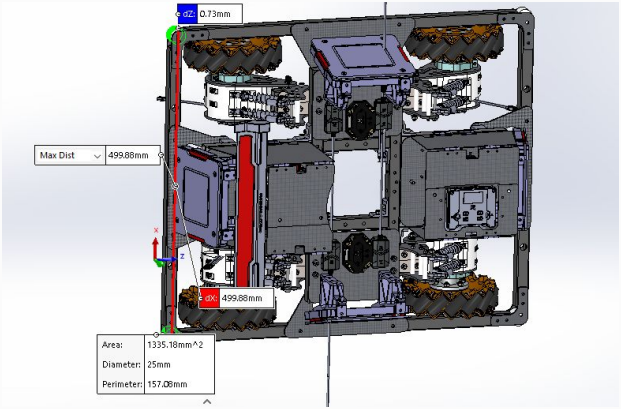
- **Turret**
  - **Capability:**
    - **Holds Ammo and Shoots Projectiles**
    - **Protection of Electronics**
    - **Servo-Powered Lid**
  - **To be improved:**
    - **Wire Management**
    - **Reduce Jamming of Projectiles**

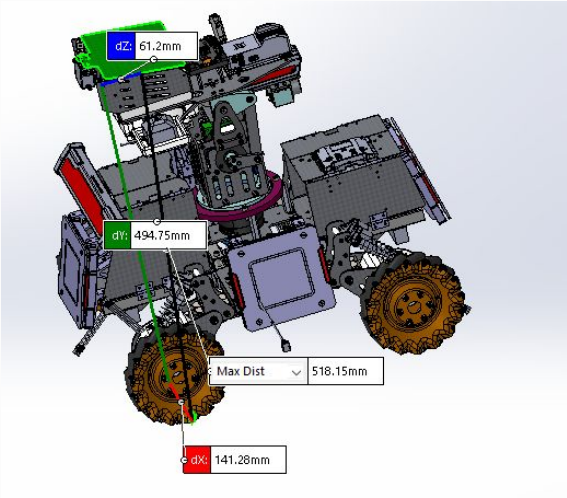


Dimensions

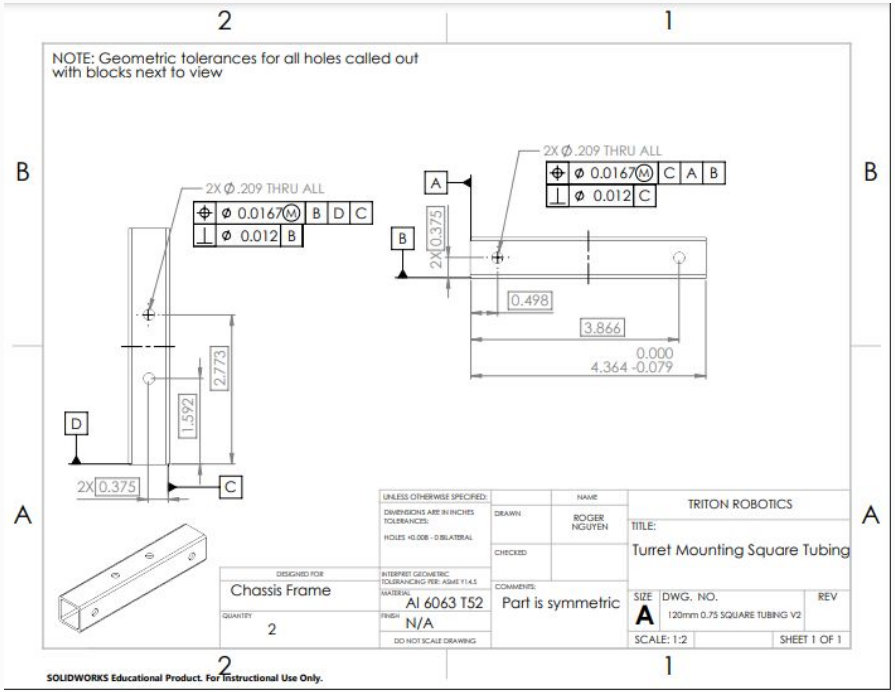
- **(L,W,H): (599.72, 499.88, 518.15) mm**



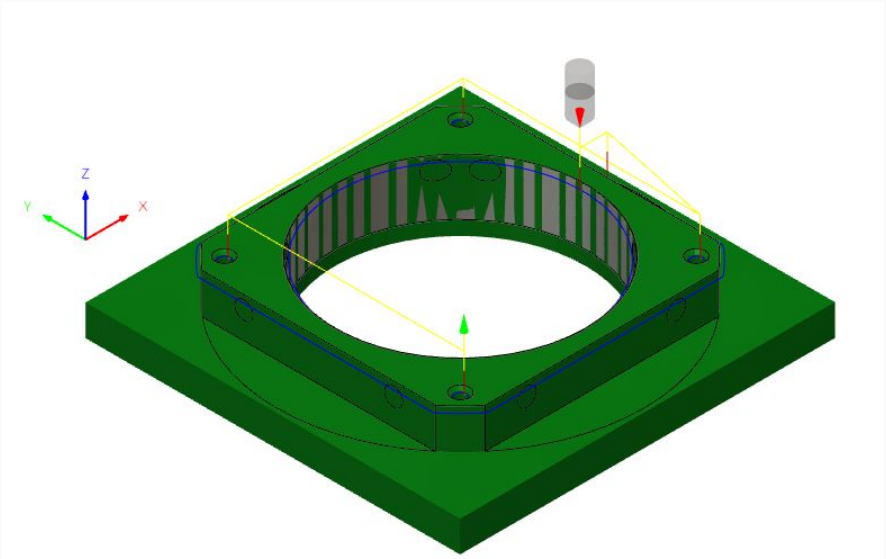




Manufacturing Showcase

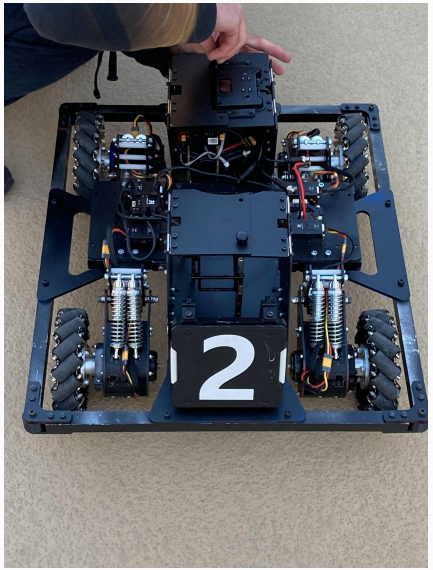


Drawing for Milling

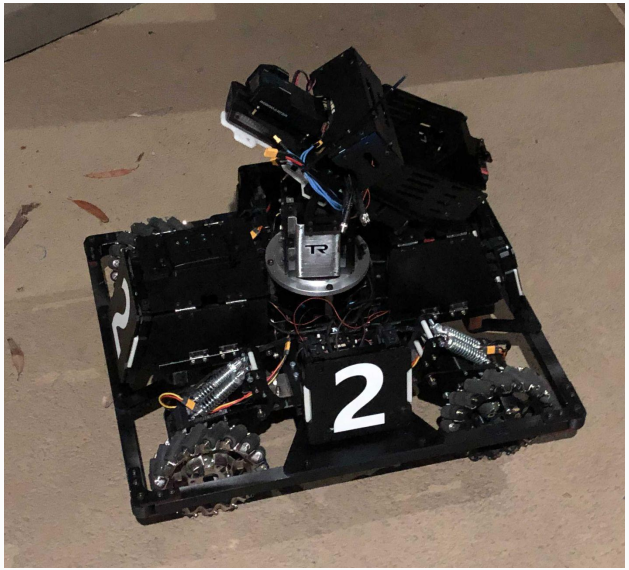


Program for CNC Milling

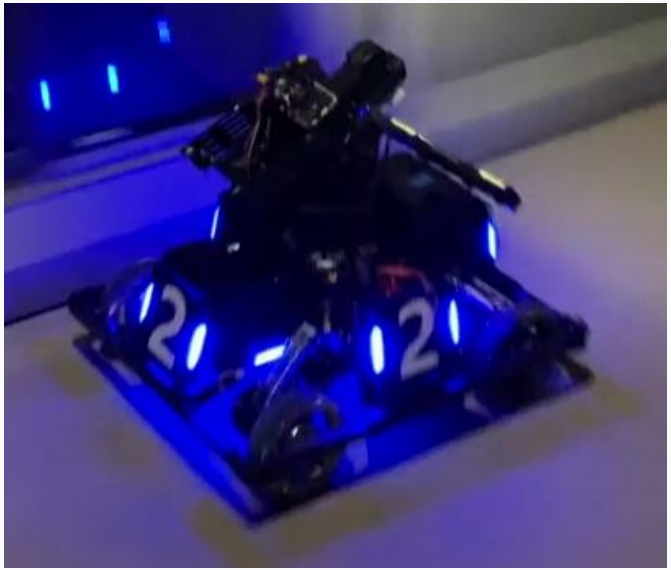
Physical Model Showcase



Robot without Turret

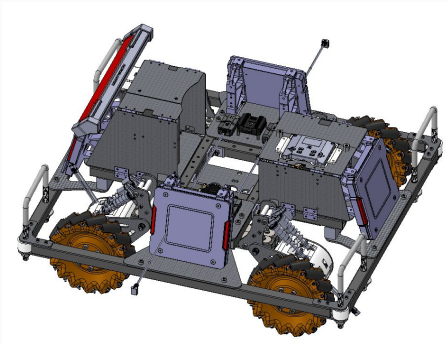


Full Robot

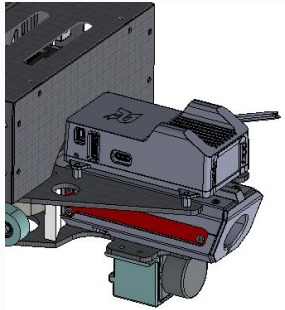


Full Robot Online

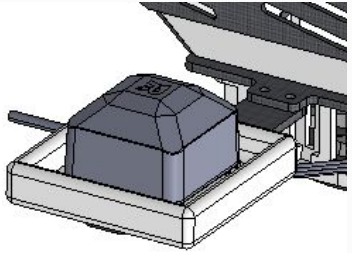
Referee System Showcase



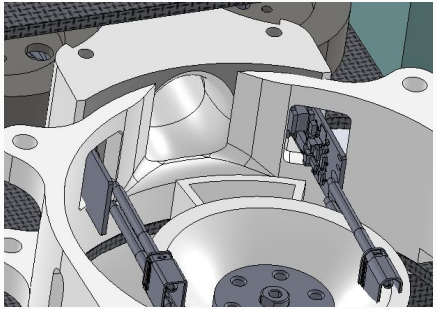
4 armor plates + 1 HP Bar



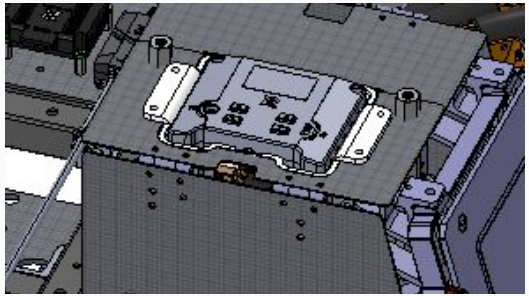
17mm Speed Monitor + VTM



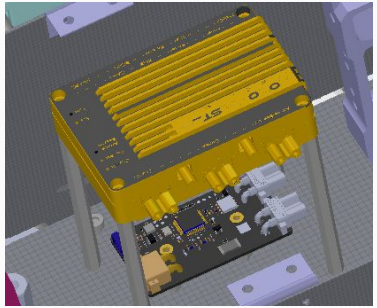
GPS



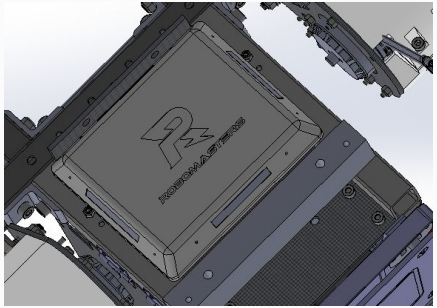
Fluorescent LED Driver



Main Control Module

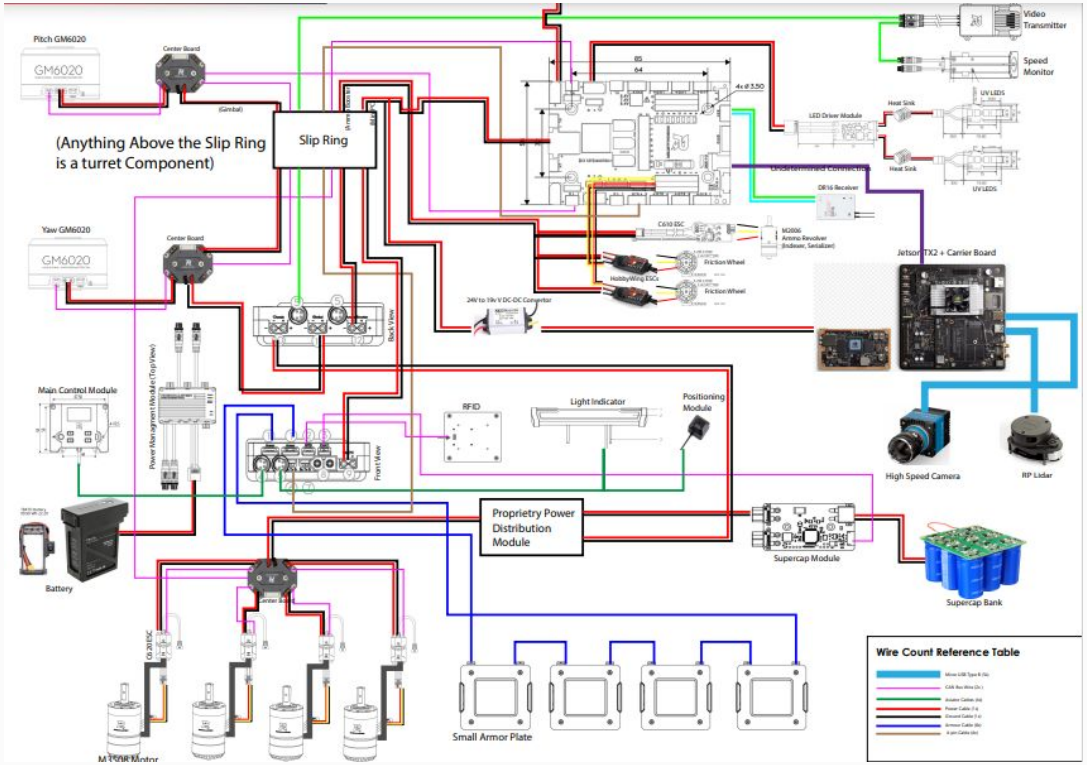


PMM + SuperCap Module



RFID

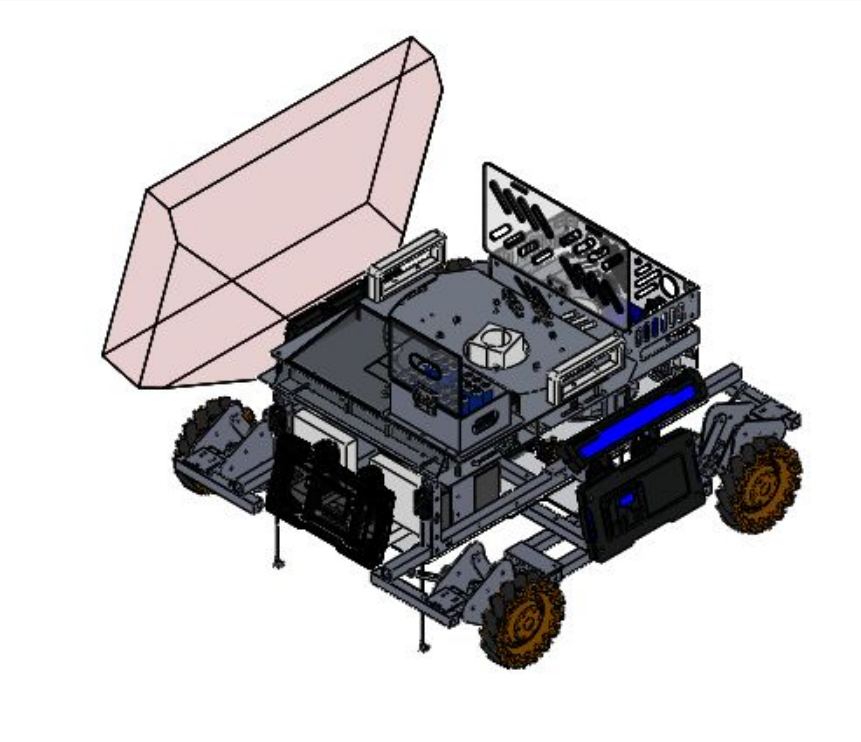
Wiring Diagram Showcase



Hero

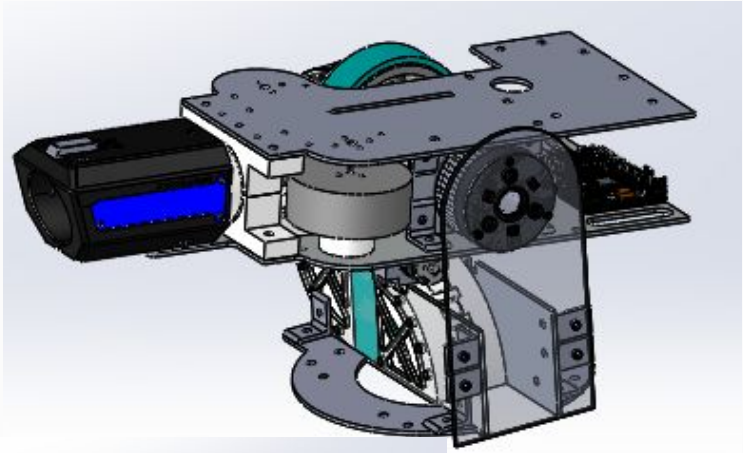
CAD Design - Chassis

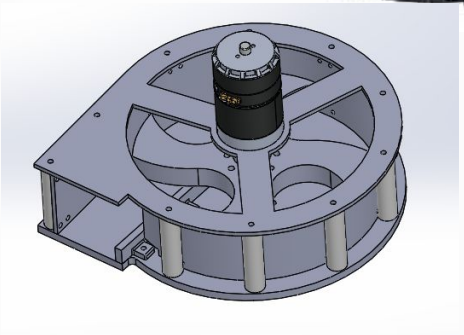
- **Chassis**
  - **Capability:**
    - **Easy access to internals**
    - **Experimental suspension design for versatile movement**
    - **Store 45 42mm rounds**
  - **To be improved:**
    - **Wiring and electronic mount**
    - **Reduce size/weight**
    - **Suspension**



CAD Design - Gimbal/ Serializer

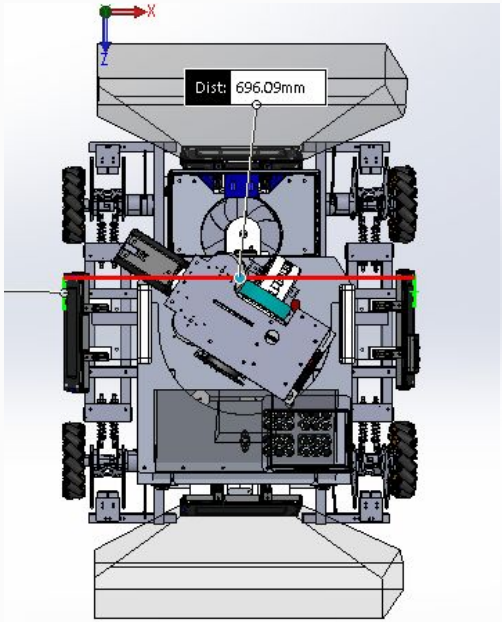
- **Gimbal**
  - **Capability:**
    - **Shoots 42 mm rounds**
    - **Bottom feed ammos**
  - **To be improved:**
    - **Add secondary 17mm turret above**
    - **Reduce friction/ increase strength of serializer (use different material)**

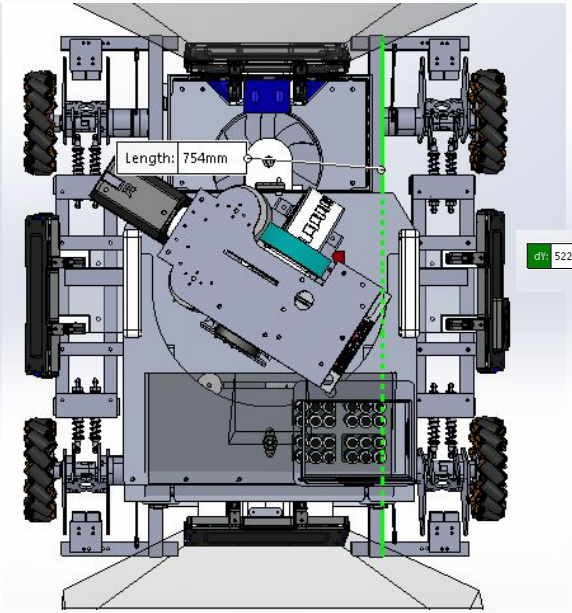


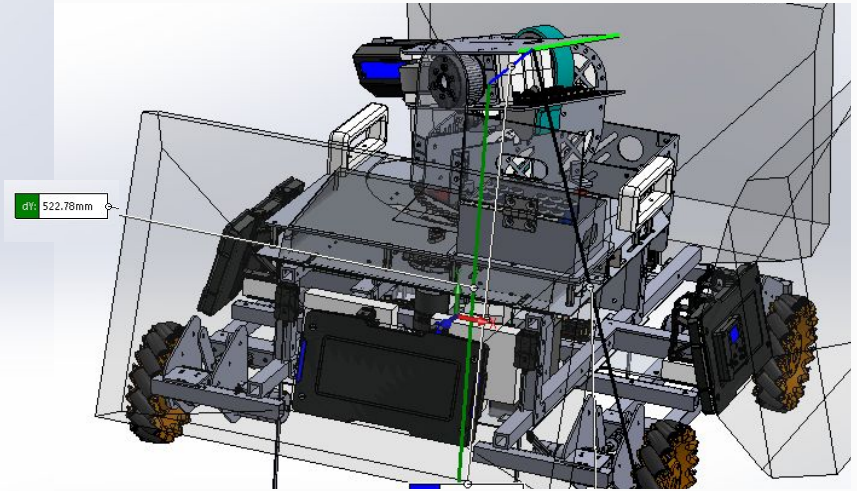


Dimensions

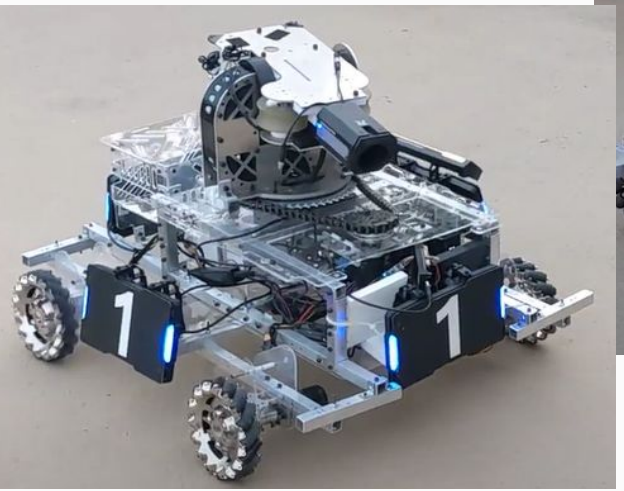
- 754x694.5x522.78 LWH

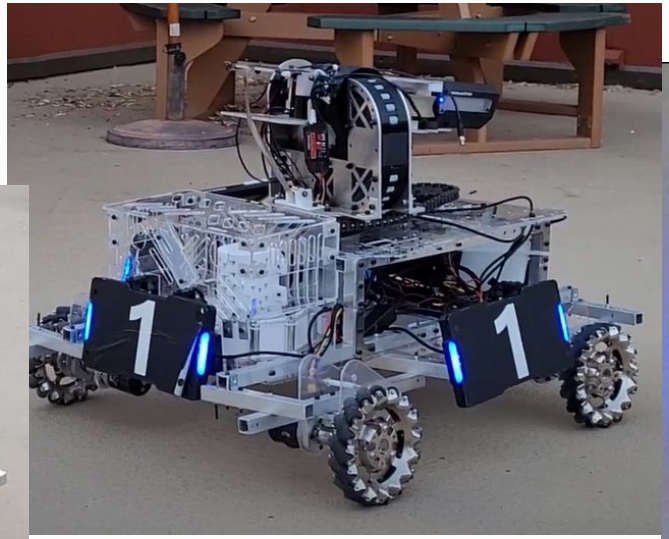


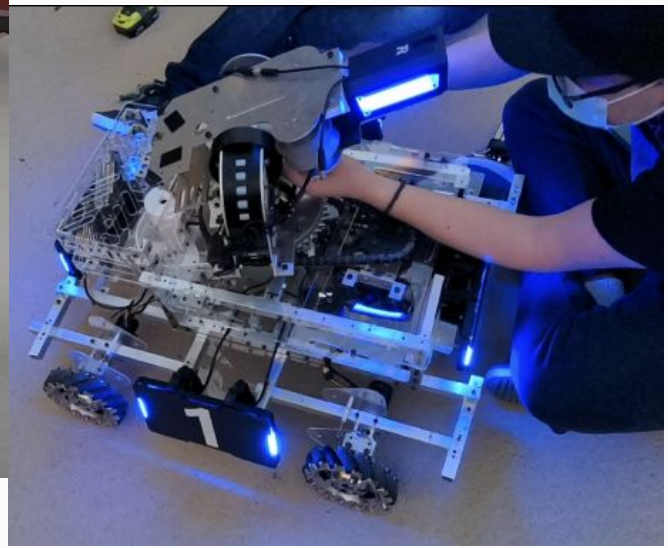




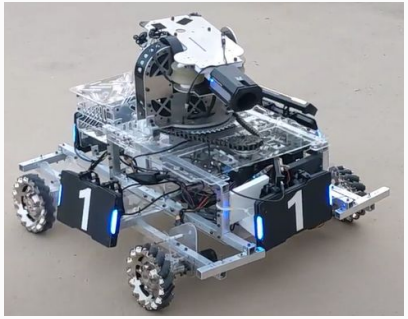
Hero Prototype

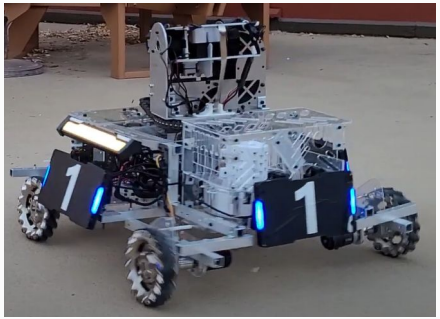




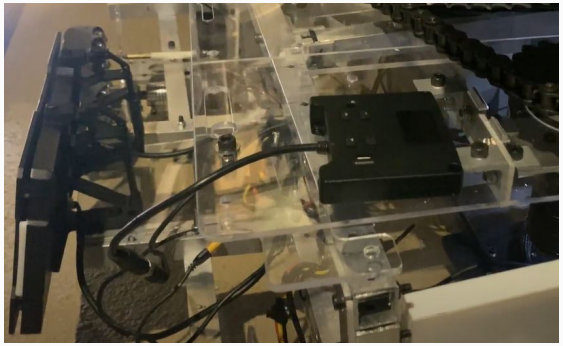


Hero Referee System





4 Large Armor Modules & Light Indicator



Main Control Module

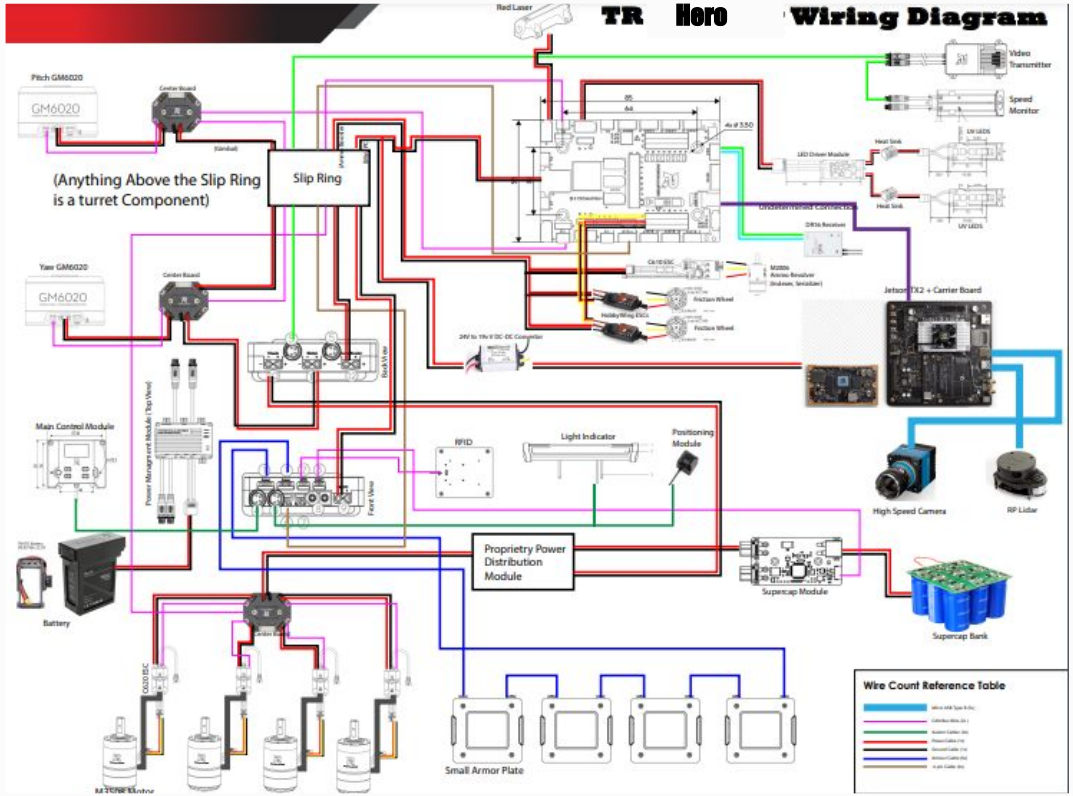
RFID Module

42mm Speed Monitor & Transmitter

Power Management Module

Positioning System mount

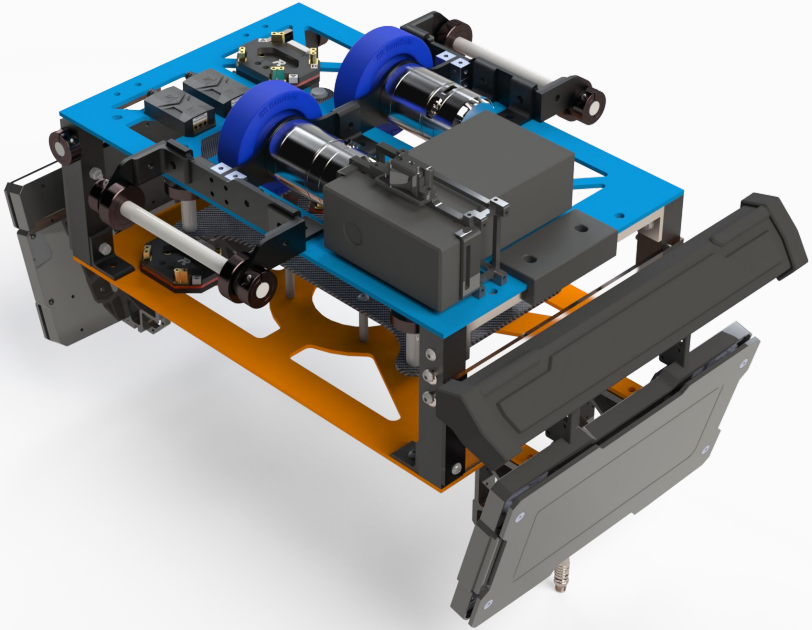
Wiring Diagram



Sentry

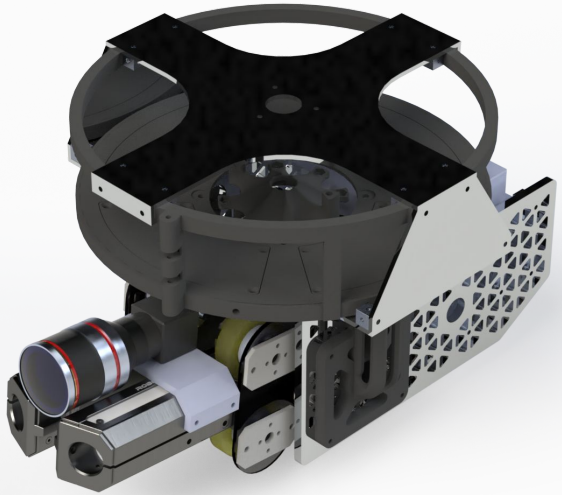
CAD Design -Chassis

- **Chassis**
  - **Capability:**
    - **Smooth movement along rail**
    - **Easy access to electronics**
    - **Easy attachment/detachment from rail**
  - **To be improved:**
    - **Side rollers need to be adjustable**



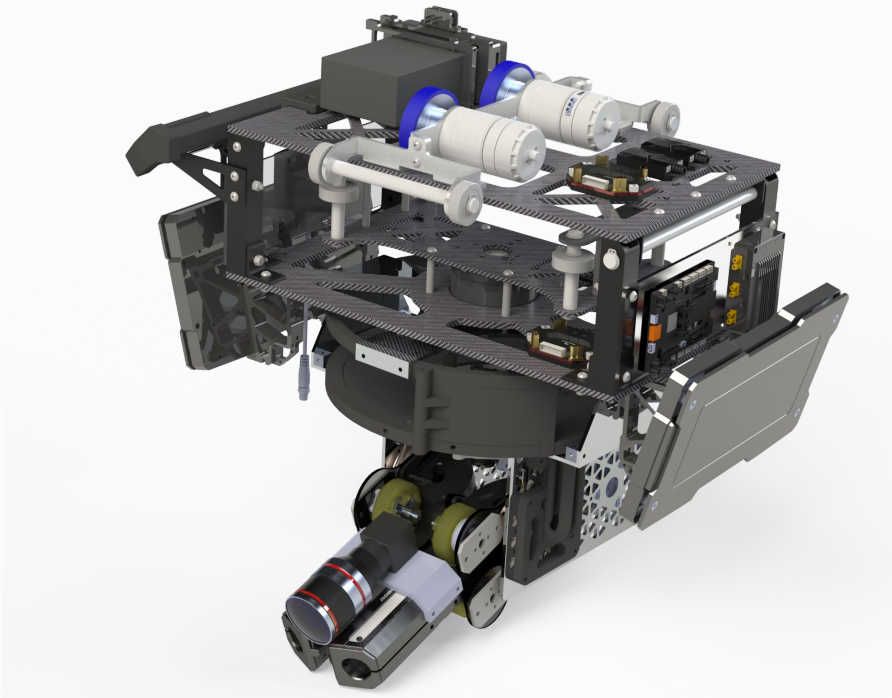
CAD Design -Turret

- **Chassis**
  - **Capability:**
    - **Dual Barrel Shooting**
    - **Compact lightweight design**
    - **Full 500 ammo capacity**
  - **To be improved:**
    - **Eliminate Indexer jamming**
    - **Ammo storage accessibility**



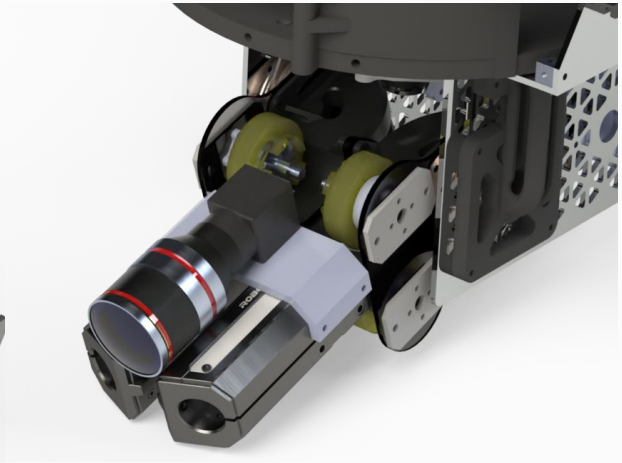
CAD Design -Sentry

- **Chassis**
  - **Capability:**
    - **Fits all sizing requirements**
    - **Fast and agile movement**
  - **To be improved:**
    - **Integration of slip ring**
  -

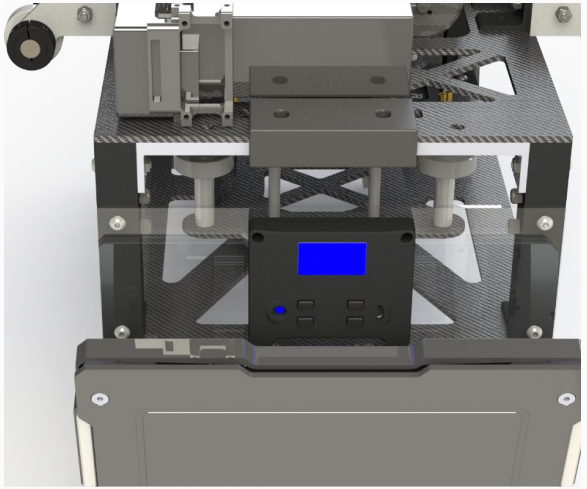


Referee System Showcase

Power Management Module



17mm Speed Monitor Module

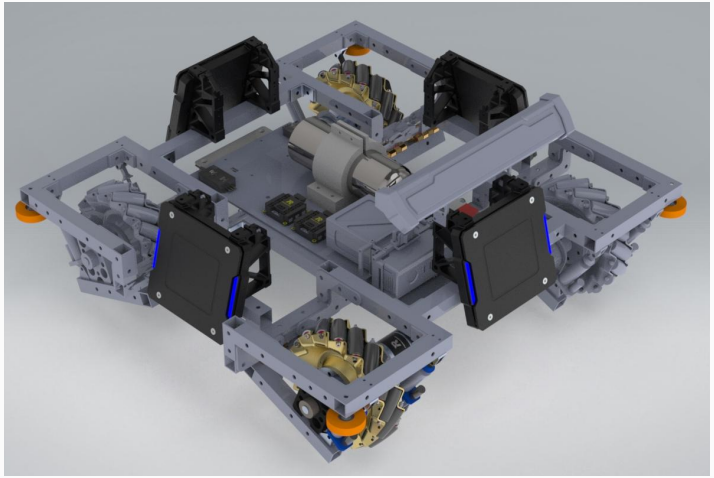


Main Control Module

Engineer

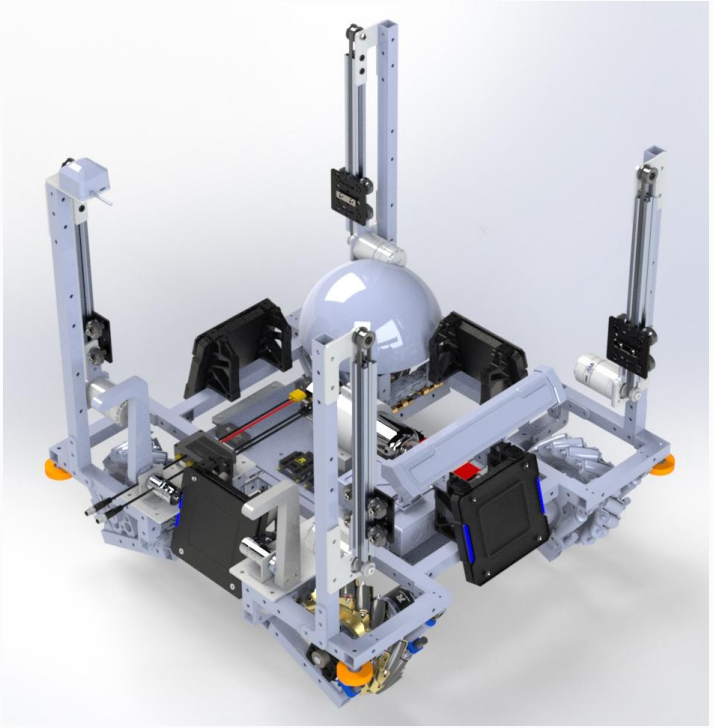
CAD Design - Chassis

- **Chassis**
  - **Capability:**
    - **Relatively stable under violent crash**
    - **Easy Access of wiring**
    - **Roller on corners of chassis to prevent stuck in arena corners**
  - **To be improved:**
    - **Side bars of suspension module are too low that may get stuck on bumpy road.**



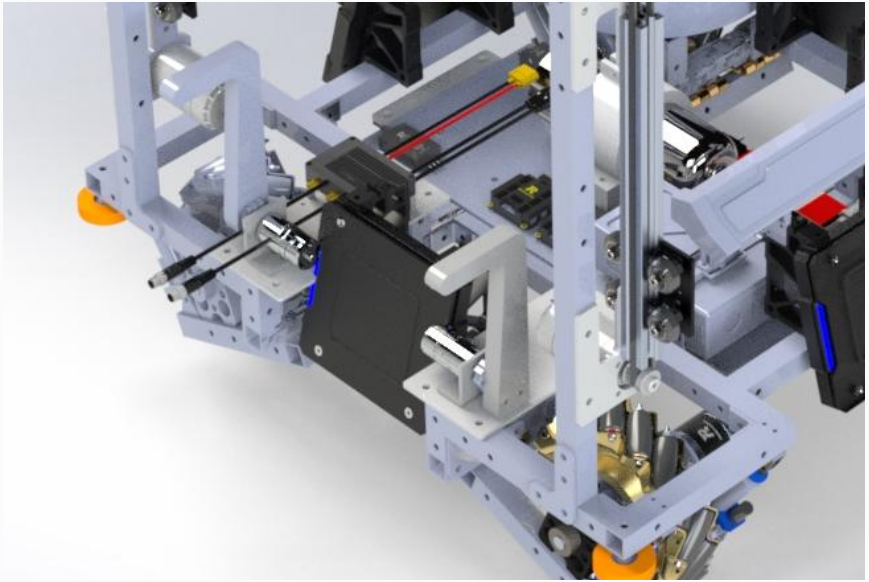
CAD Design - Lifting Mechanism

- **Lifting Mechanism**
  - **Capability:**
    - **Can move 200mm vertically which covers the needs in competition**
  - **To be improved:**
    - **More connections between the 4 vertical modules to improve stability**
    - **Add mobility in horizontal direction**



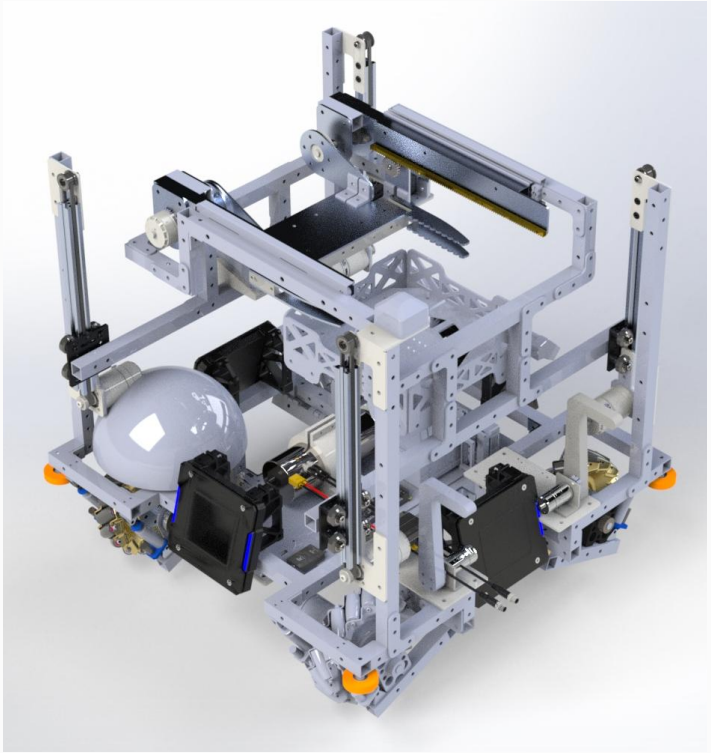
CAD Design - Rescue Mechanism (not tested)

- **2DMG**
  - **Capability:**
    - **Drag protective case of infantry and hero**
  - **To be improved:**
    - **Use one motor with shaft or pneumatic piston**

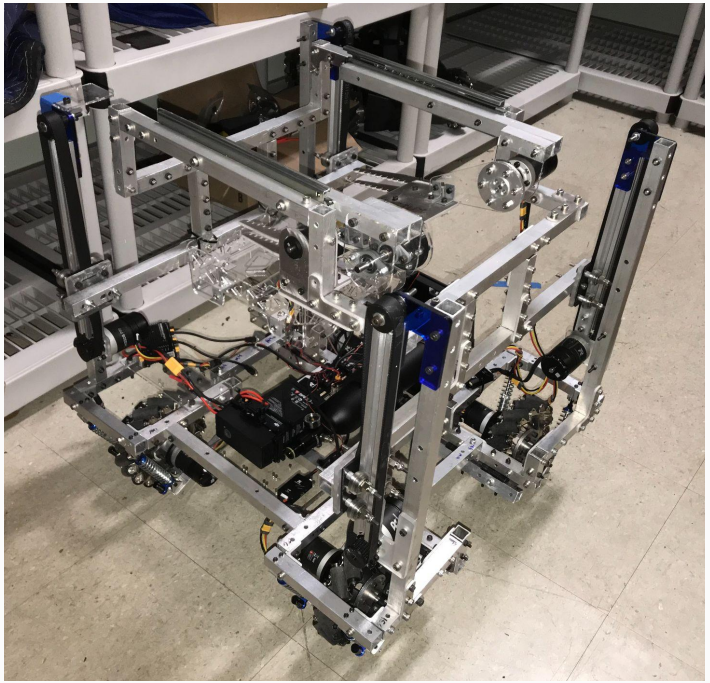


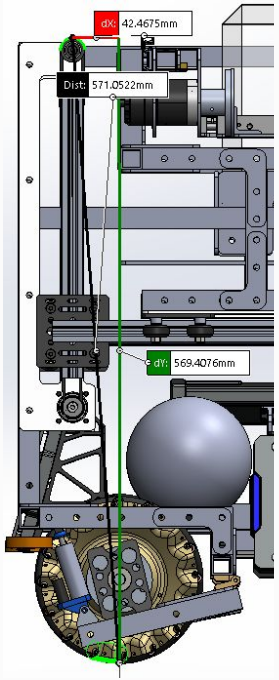
CAD Design - Claw and Mineral Box

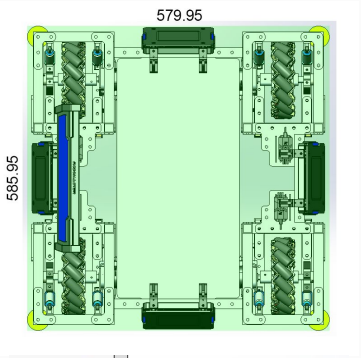
- **Claw and Mineral Box**
  - **Capability:**
    - **Extend 400mm**
    - **Store one mineral in the rear**
  - **To be improved:**
    - **Adding space between the sides of the claw to have more tolerance for flipping back**
    - **Close the grippers tighter**

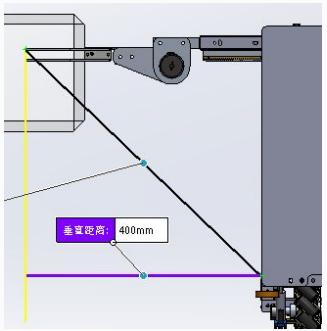


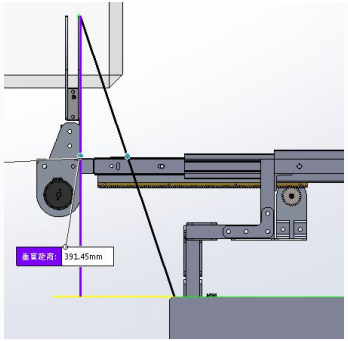
Engineer Prototype

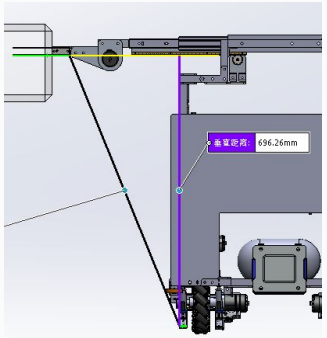




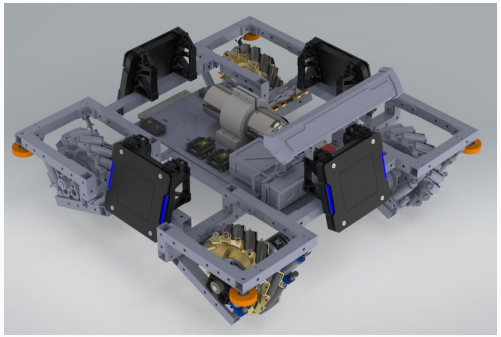




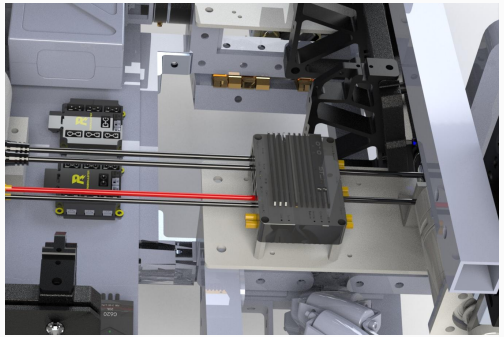




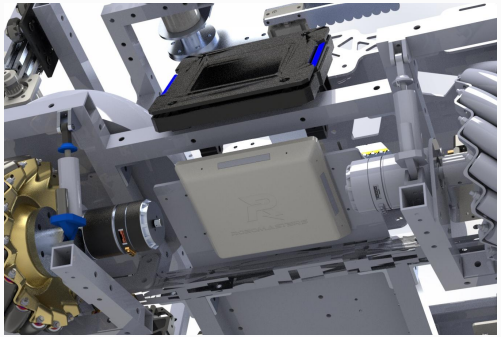
Referee System Showcase



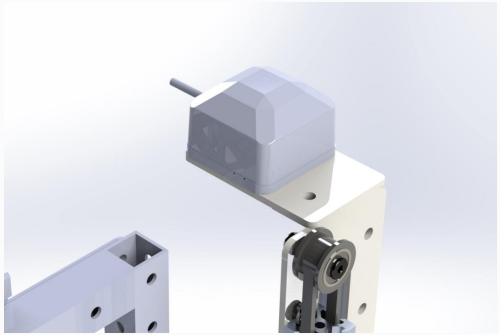
4 armor plates + 1 HP Bar



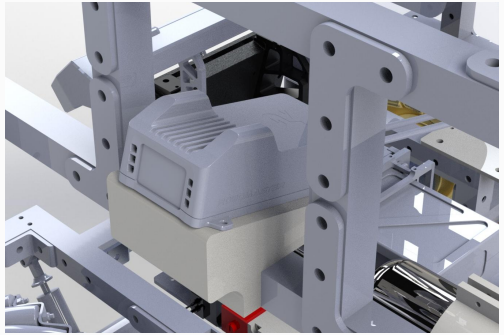
Power Management Module



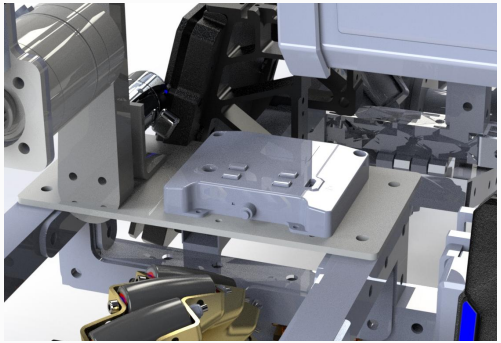
RFID



GPS

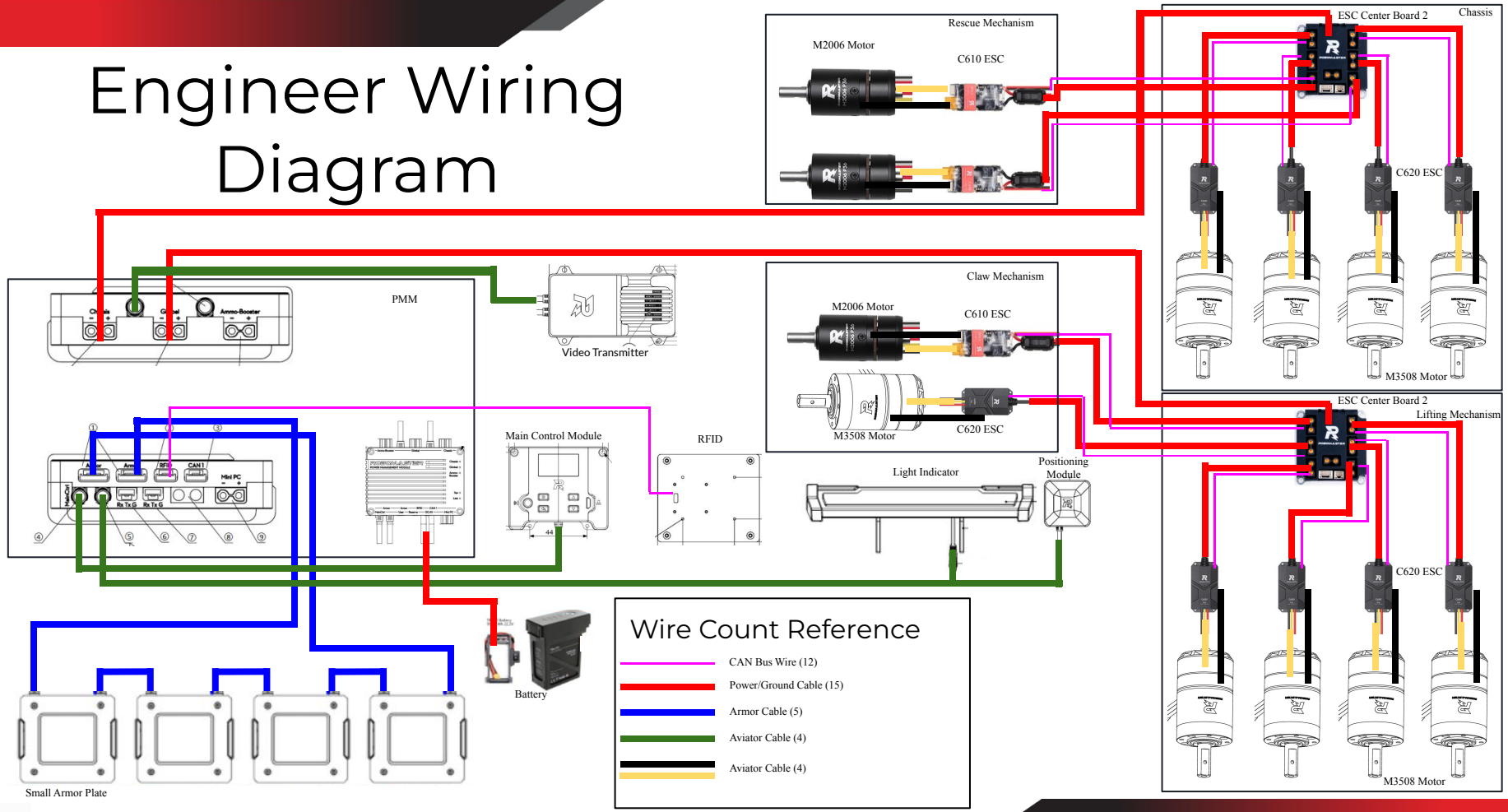


VTM



Main Control Module

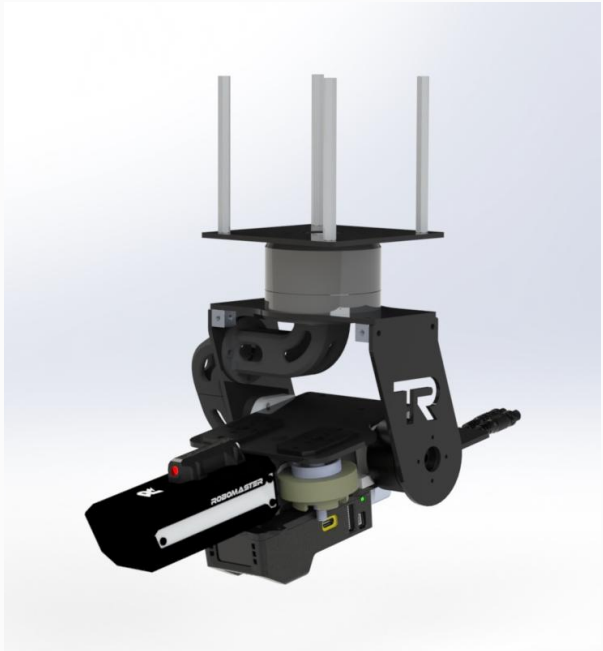
Engineer Wiring Diagram



Drone

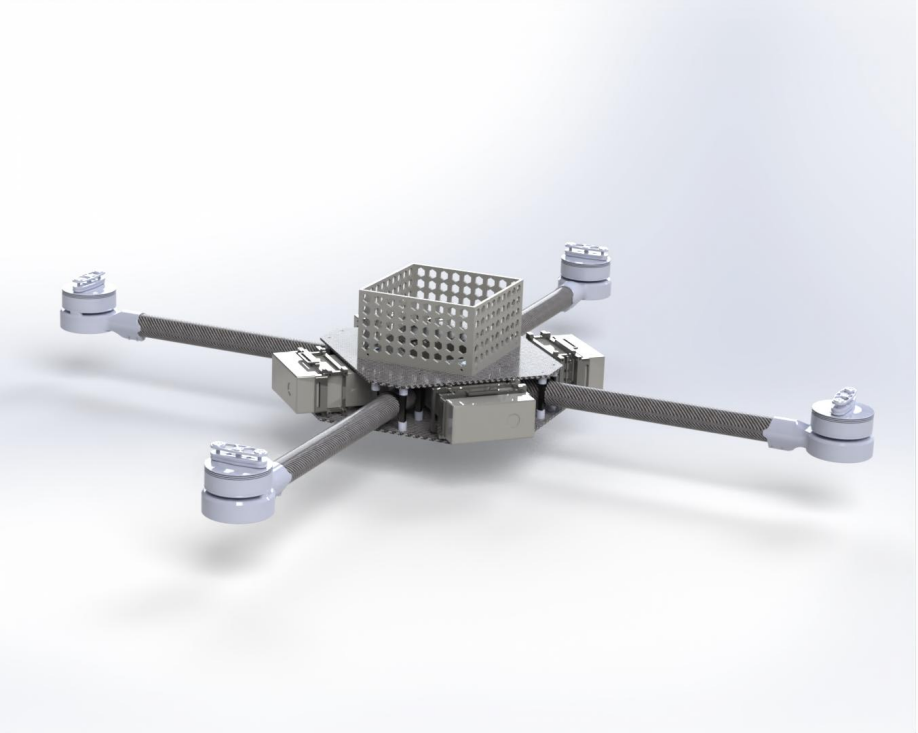
CAD Design - Turret

- **Turret**
  - **Capability:**
    - **Shoot 17mm balls at 25 m/s**
    - **Allows for easy dev board access**
  - **To be improved:**
    - **Not a lot of space to mount the mini PC**
    - **Needs better wire management**



CAD Design - Airframe

- **Airframe**
  - **Capability:**
    - **Flight time of about 20 minutes**
  - **To be improved:**
    - **Needs some more mounting holes**
    - **Needs better wire management**

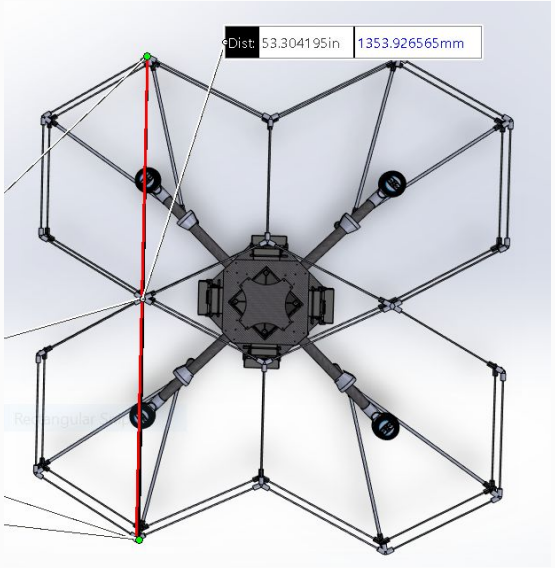


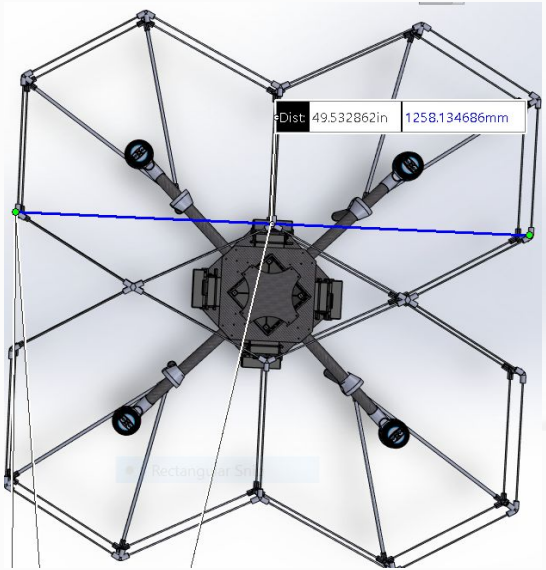
CAD Design - Cage and Landing Gear

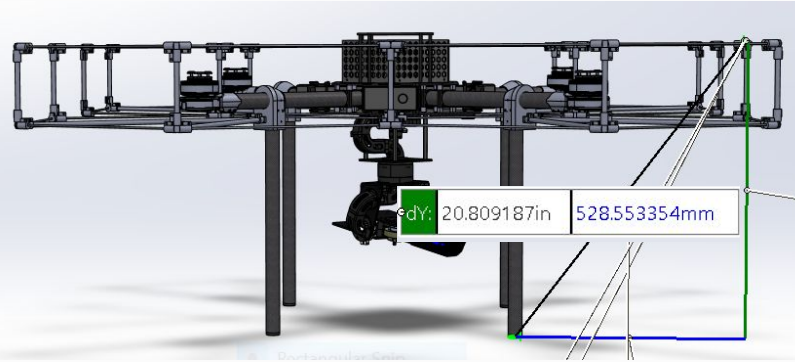
- **Cage and Landing Gear**
  - **Capability:**
    - **Can deflect balls away from propellers**
    - **Protect propellers in case of a crash**
  - **To be improved:**
    - **Landing gear legs are not stable**
    - **Cage needs more secure mounting to the airframe**

Dimensions

- 1353x1258x529mm LWH



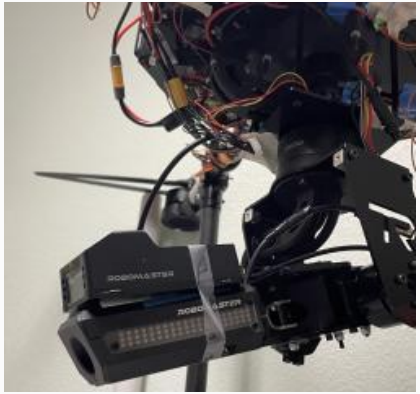




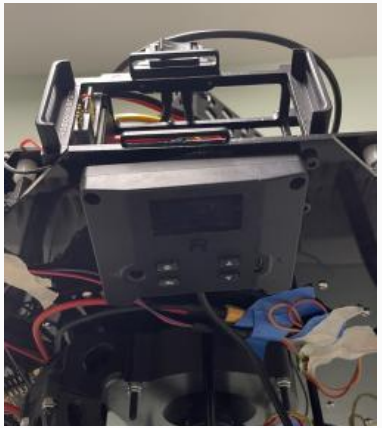
Drone Prototype

Referee System Showcase

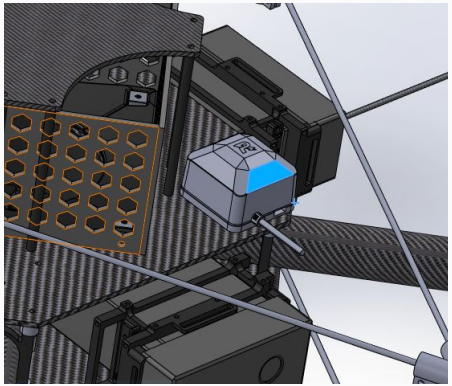
Power Management Module



17mm Speed Monitor Module and VTM



Main Control Module

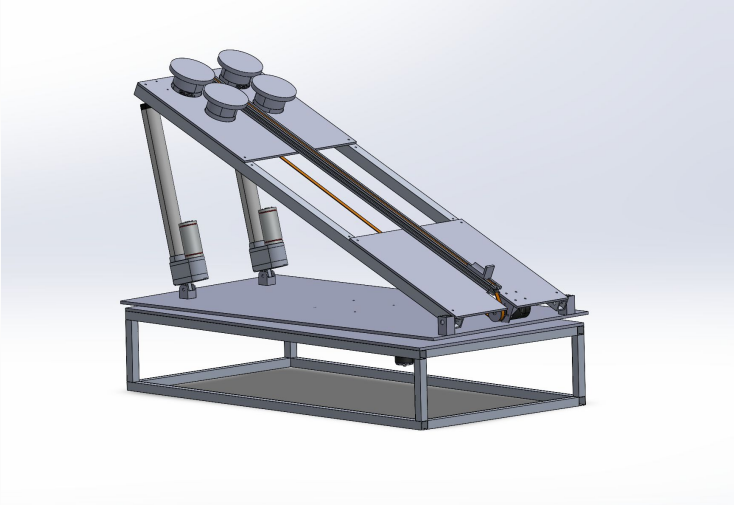


GPS

Dart

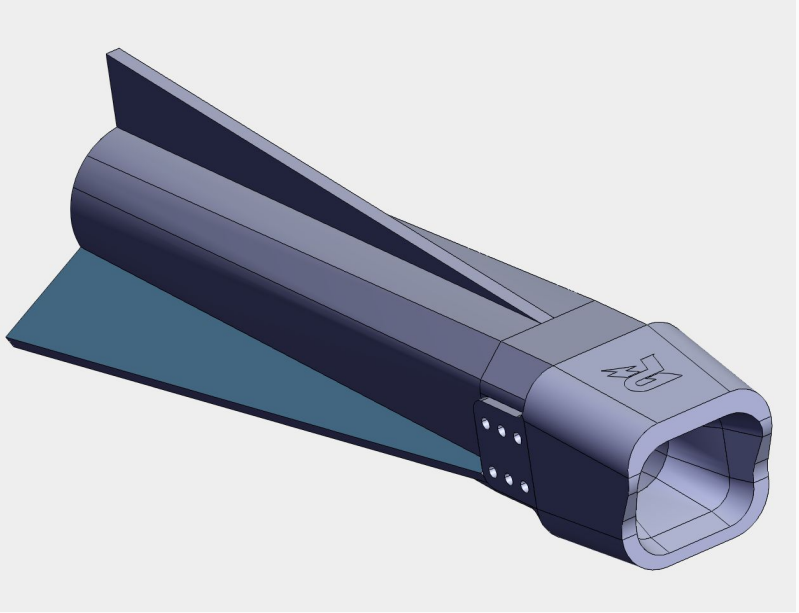
CAD Design - Launcher

- **Dart Launcher**
  - **Capabilities**
    - **Pitch movement**
    - **Yaw rotation**
    - **Movement of Dart along linear path**
    - **Shooting Dart from flywheels**
  - **To be Improved**
    - **Testing of dart launcher**
    - **Speed of all mechanisms**



CAD Design - Dart

- **Dart**
  - **Capabilities**
    - **Emit lazer light from tip**
    - **Fly with some accuracy to hit target**
  - **To be Improved**
    - **Testing of dart**
    - **Aerodynamics of dart**
    - **Self Guidance features**



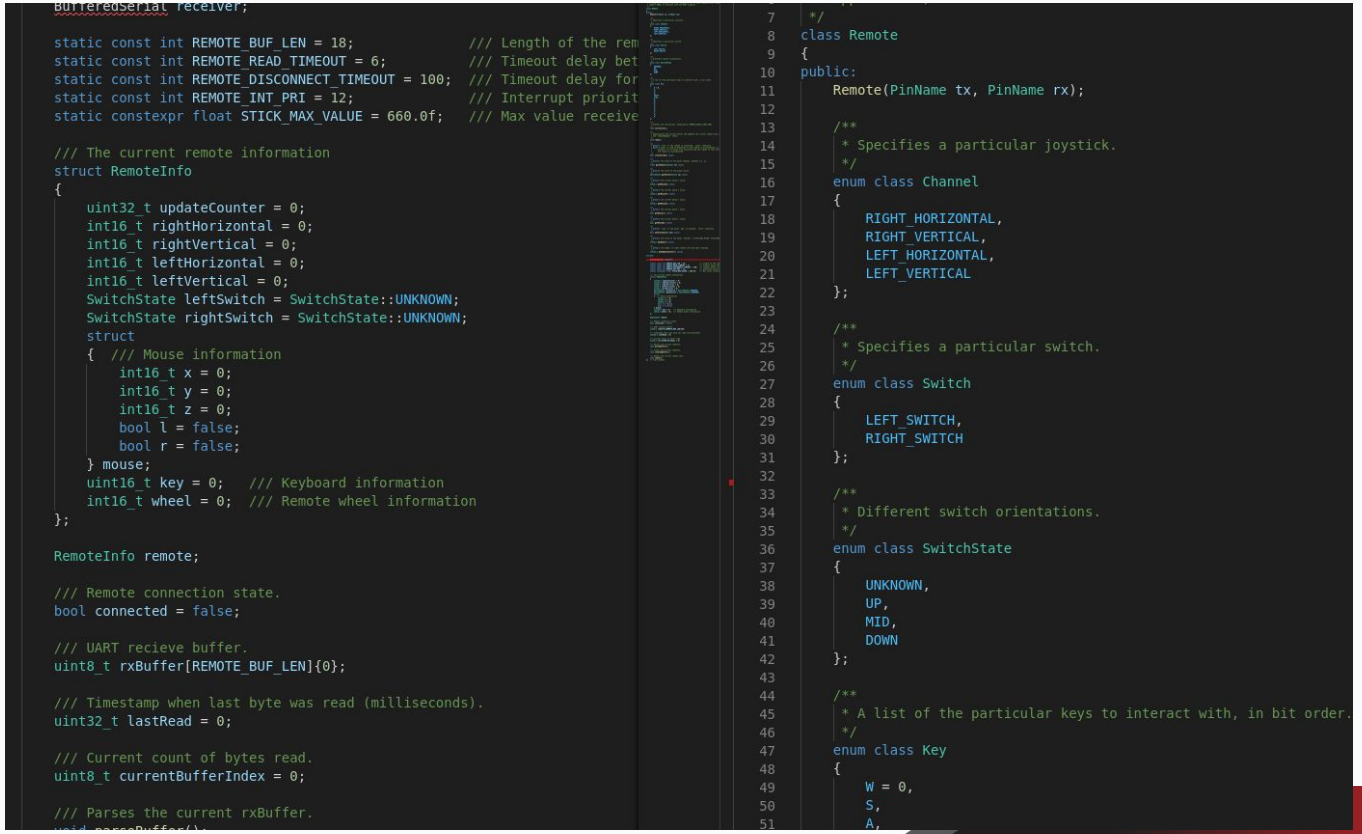
Embedded

General - Remote Control

```cpp
void Remote::read()
{
    // Read next byte if available and more needed for the current packet
    //printf("Attempting to read \n");
    receiver.read(rxBuffer, sizeof(rxBuffer));

    for (int i = 0; i < sizeof(rxBuffer); i++) {
        rxBuffer[i] = ~rxBuffer[i];
        //printf("%d\t", rxBuffer[i]);
    }
    //printf("\n");
    //printf("Parsing buffer\n");
    parseBuffer();
    clearRxBuffer();

}

bool Remote::isConnected() const { return connected; }

float Remote::getChannel(Channel ch) const
{
    switch (ch)
    {
        case Channel::RIGHT_HORIZONTAL:
            return remote.rightHorizontal / STICK_MAX_VALUE;
        case Channel::RIGHT_VERTICAL:
            return remote.rightVertical / STICK_MAX_VALUE;
        case Channel::LEFT_HORIZONTAL:
            return remote.leftHorizontal / STICK_MAX_VALUE;
        case Channel::LEFT_VERTICAL:
            return remote.leftVertical / STICK_MAX_VALUE;
    }
    return 0;
}

Remote::SwitchState Remote::getSwitch(Switch sw) const
{
    switch (sw)
    {
        case Switch::LEFT_SWITCH:
            return remote.leftSwitch;
        case Switch::RIGHT_SWITCH:
            return remote.rightSwitch;
    }
    return SwitchState::UNKNOWN;
```

```cpp
    {
        return;
    }

    // mouse input
    remote.mouse.x = rxBuffer[6] | (rxBuffer[7] << 8);    // x axis
    remote.mouse.y = rxBuffer[8] | (rxBuffer[9] << 8);    // y axis
    remote.mouse.z = rxBuffer[10] | (rxBuffer[11] << 8);  // z axis
    remote.mouse.l = static_cast<bool>(rxBuffer[12]);     // left button cli
    remote.mouse.r = static_cast<bool>(rxBuffer[13]);     // right button cl

    // keyboard capture
    remote.key = rxBuffer[14] | rxBuffer[15] << 8;
    // Remote wheel
    remote.wheel = (rxBuffer[16] | rxBuffer[17] << 8) - 1024;

    remote.updateCounter++;
}

void Remote::clearRxBuffer()
{
    // Reset bytes read counter
    currentBufferIndex = 0;
    // Clear remote rxBuffer
    for (int i = 0; i < REMOTE_BUF_LEN; i++)
    {
        rxBuffer[i] = 0;
    }
    // Clear Usart1 rxBuffer

}

void Remote::reset()
{
    remote.rightHorizontal = 0;
    remote.rightVertical = 0;
    remote.leftHorizontal = 0;
    remote.leftVertical = 0;
    remote.leftSwitch = SwitchState::UNKNOWN;
    remote.rightSwitch = SwitchState::UNKNOWN;
    remote.mouse.x = 0;
    remote.mouse.y = 0;
    remote.mouse.z = 0;
    remote.mouse.l = 0;
    remote.mouse.r = 0;
    remote.key = 0;
```

General - Remote Control



```
BufferedSerial receiver;

static const int REMOTE_BUF_LEN = 18;              /// Length of the rem
static const int REMOTE_READ_TIMEOUT = 6;          /// Timeout delay bet
static const int REMOTE_DISCONNECT_TIMEOUT = 100;  /// Timeout delay for
static const int REMOTE_INT_PRI = 12;              /// Interrupt priorit
static constexpr float STICK_MAX_VALUE = 660.0f;   /// Max value receive

/// The current remote information
struct RemoteInfo
{
    uint32_t updateCounter = 0;
    int16_t rightHorizontal = 0;
    int16_t rightVertical = 0;
    int16_t leftHorizontal = 0;
    int16_t leftVertical = 0;
    SwitchState leftSwitch = SwitchState::UNKNOWN;
    SwitchState rightSwitch = SwitchState::UNKNOWN;
    struct
    {  /// Mouse information
        int16_t x = 0;
        int16_t y = 0;
        int16_t z = 0;
        bool l = false;
        bool r = false;
    } mouse;
    uint16_t key = 0;   /// Keyboard information
    int16_t wheel = 0;  /// Remote wheel information
};

RemoteInfo remote;

/// Remote connection state.
bool connected = false;

/// UART recieve buffer.
uint8_t rxBuffer[REMOTE_BUF_LEN]{0};

/// Timestamp when last byte was read (milliseconds).
uint32_t lastRead = 0;

/// Current count of bytes read.
uint8_t currentBufferIndex = 0;

/// Parses the current rxBuffer.
```

```
 */
class Remote
{
public:
    Remote(PinName tx, PinName rx);

    /**
     * Specifies a particular joystick.
     */
    enum class Channel
    {
        RIGHT_HORIZONTAL,
        RIGHT_VERTICAL,
        LEFT_HORIZONTAL,
        LEFT_VERTICAL
    };

    /**
     * Specifies a particular switch.
     */
    enum class Switch
    {
        LEFT_SWITCH,
        RIGHT_SWITCH
    };

    /**
     * Different switch orientations.
     */
    enum class SwitchState
    {
        UNKNOWN,
        UP,
        MID,
        DOWN
    };

    /**
     * A list of the particular keys to interact with, in bit order.
     */
    enum class Key
    {
        W = 0,
        S,
        A,
```

General - Servo

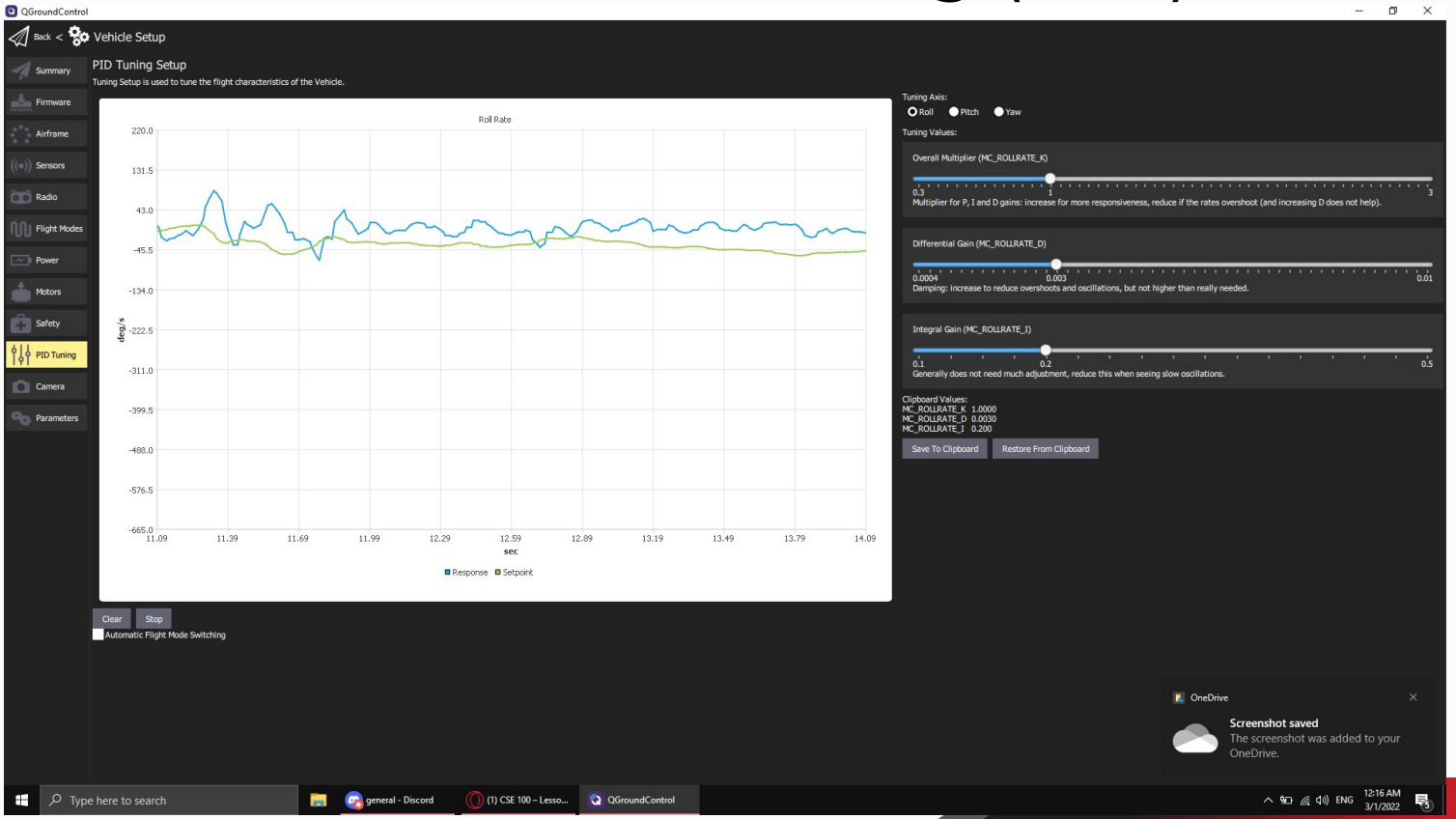
```cpp
#include "mbed.h"
#include "../include/servo.h"

Servo::Servo(PinName pin) : pwm(pin) {
    pwm.period_ms(20);
    Servo::inverted = false;
}

void Servo::set(int pos){

    if (pos < -60){
        Servo::position = -60;
    }
    else if (pos > 60){
        Servo::position = 60;
    }
    else{
        Servo::position = pos;
    }

    if (Servo::inverted){
        pwm.pulsewidth_us(1500 - Servo::position * 10);
    }
    else {
        pwm.pulsewidth_us(1500 + Servo::position * 10);
    }

}

int Servo::get(){
    return Servo::position;
}

void Servo::invert(bool inv){
    Servo::inverted = inv;
}
```

```cpp
#include "mbed.h"

class Servo {
    public:
        Servo(PinName pin);
        void set(int pos);
        int get();
        void invert(bool inv);
    private:
        PwmOut pwm;
        int position;
        bool inverted;
};
```

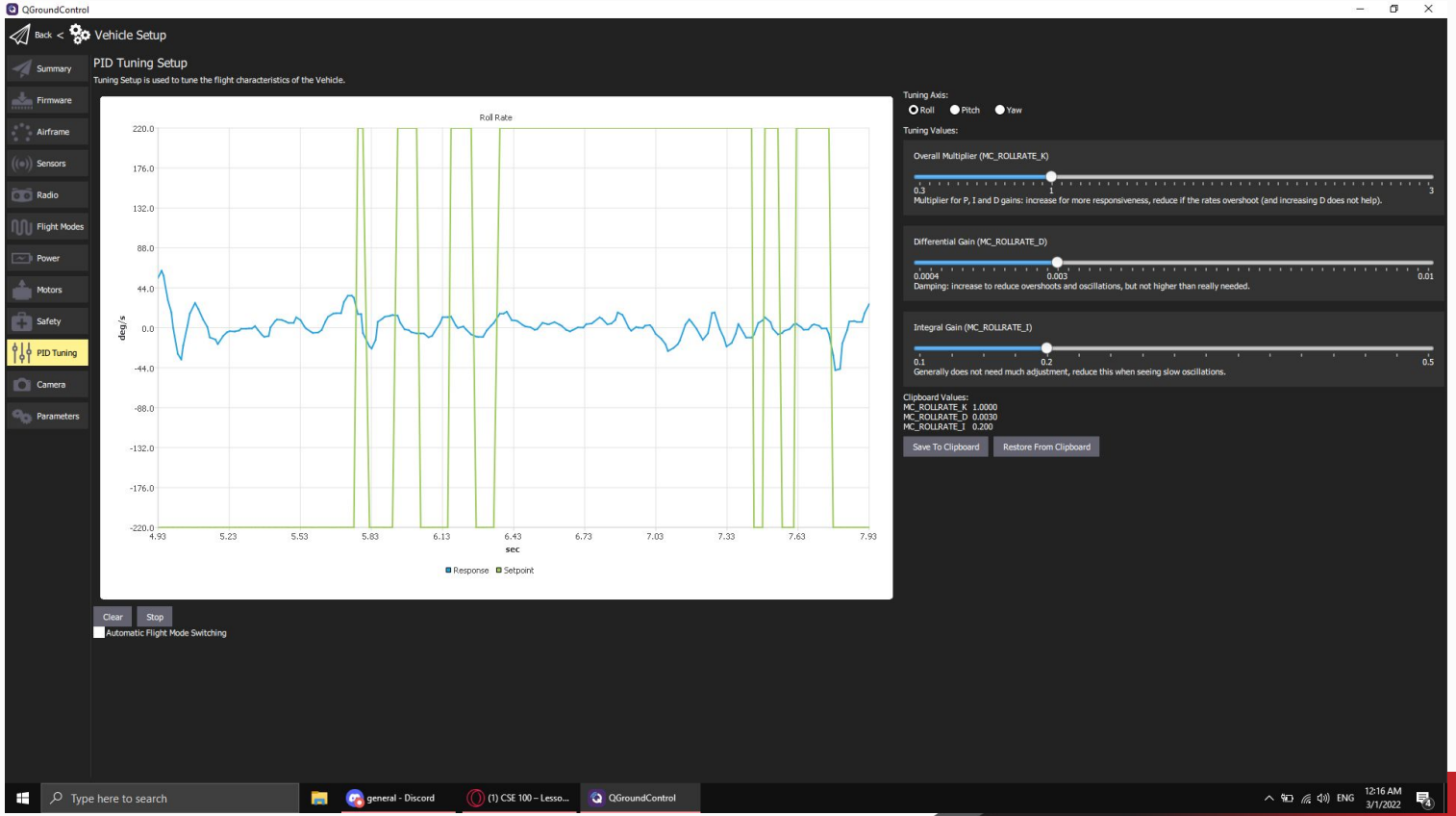
Infantry - BeyBlade Mode

```
        case 2: // reserve for keyboard
        default: // when remote not connected
                break;

}
sig->gimbal.pitch_ecd += (short)((joyRightY * JOY_TO_ECD) * 0.05f); // pitch don't have the same
float raw_x = joyLeftX * JOY_TO_RPM * DRIVE_SPEED_DEFAULT;
float raw_y = joyLeftY * JOY_TO_RPM * DRIVE_SPEED_DEFAULT;

// change of angle from the original position
radian = sig->gimbal.yaw_ecd / ECD_PERIOD * 2 * PI;

// apply rotational matrix
x = (float) (raw_x * cos(radian) - raw_y * sin(radian));
y = (float) (raw_x * sin(radian) + raw_y * cos(radian));

sig->chassis.LF_rpm = (short) ((y + x + baybladeRotation) * RPM_SCALE);
sig->chassis.RF_rpm = (short) ((-y + x + baybladeRotation) * RPM_SCALE);
sig->chassis.LB_rpm = (short) ((y - x + baybladeRotation) * RPM_SCALE);
sig->chassis.RB_rpm = (short) ((-y - x + baybladeRotation) * RPM_SCALE);
```

Drone - PID Tuning (Roll)

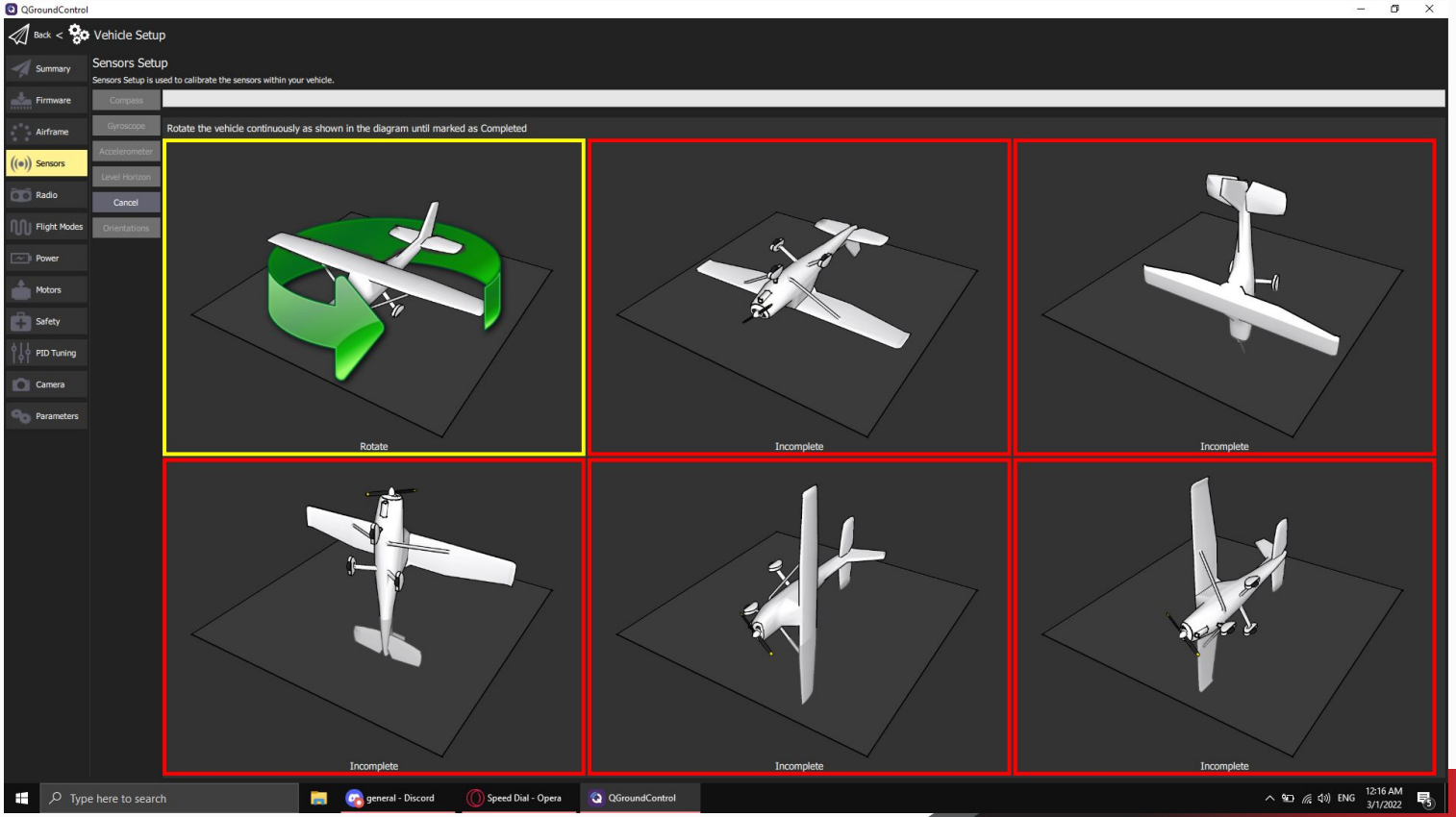


Drone - PID Tuning (Roll)



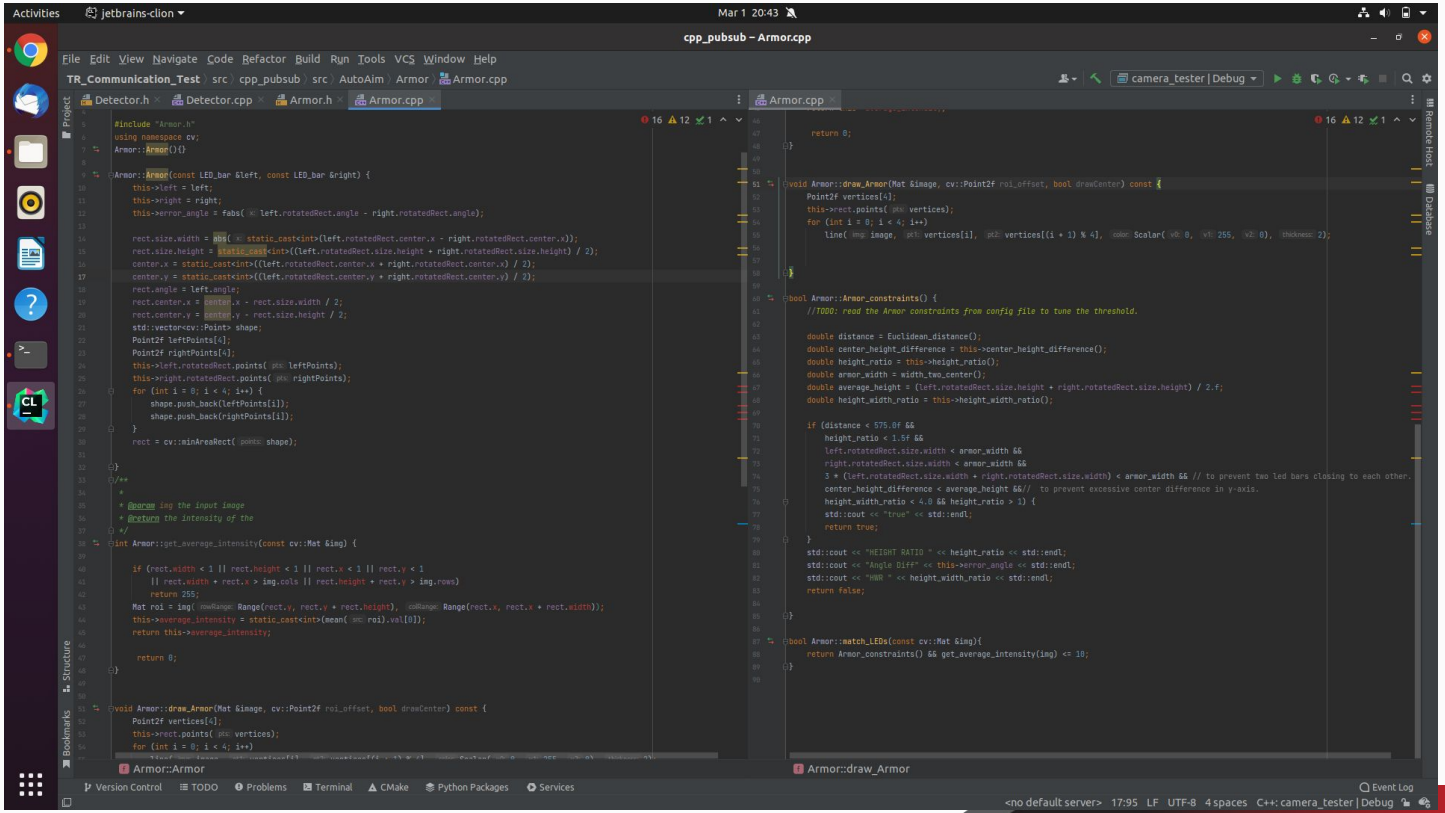
Drone - PID Tuning (Yaw)

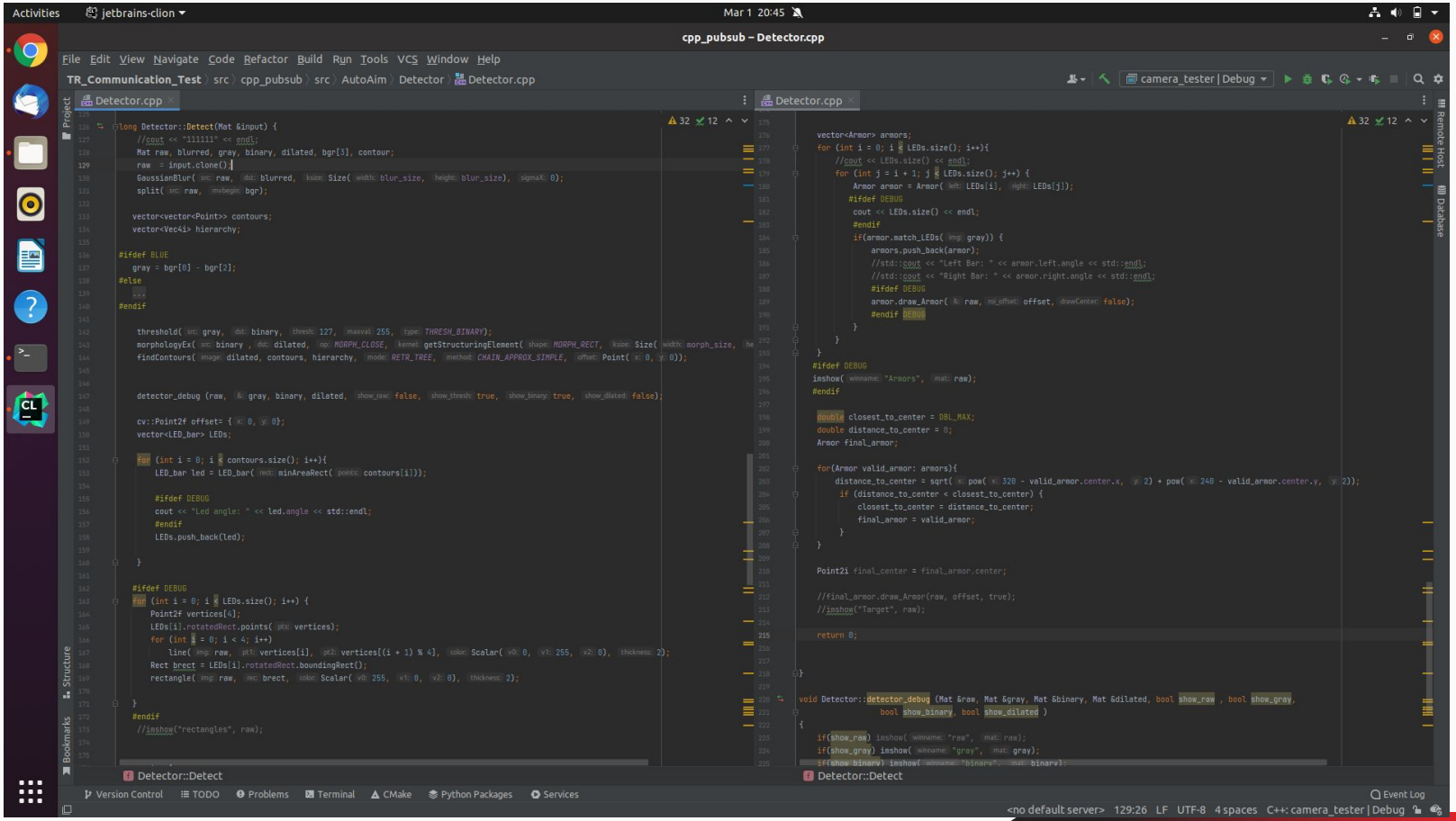
Drone - Sensor



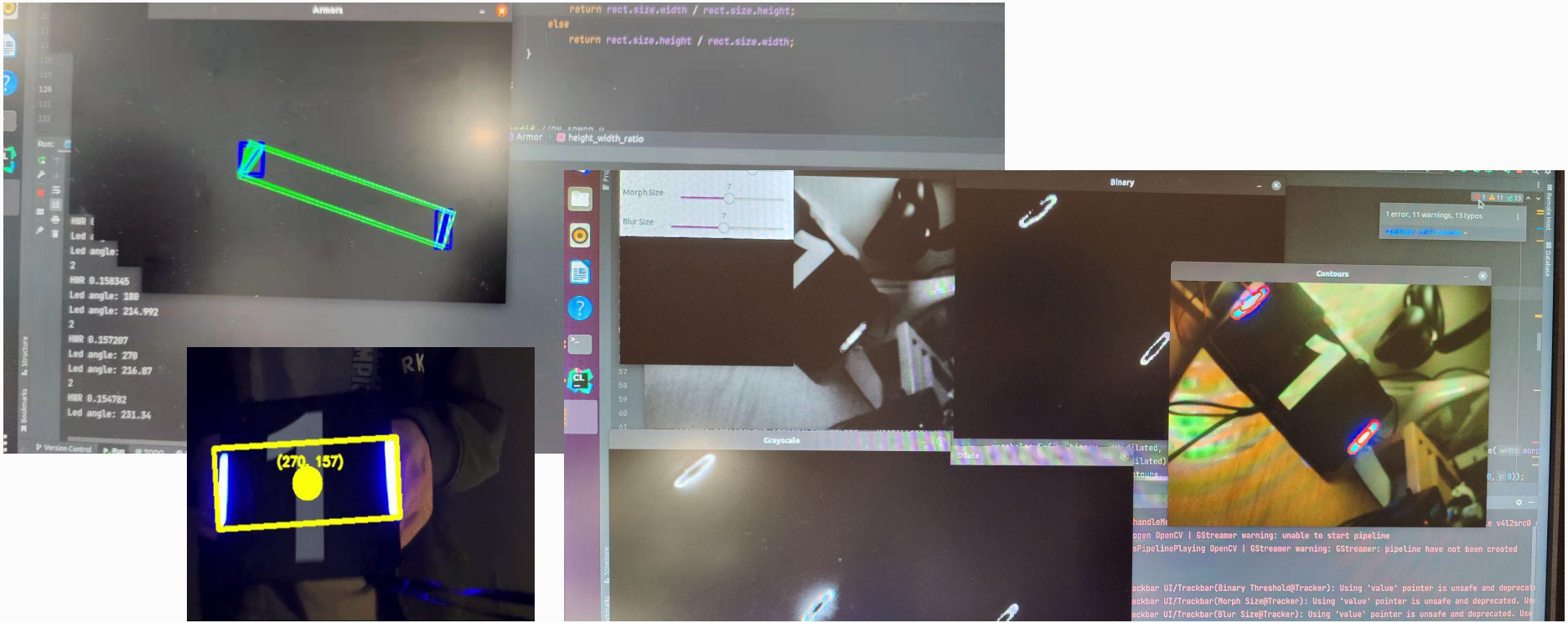
Computer Vision

Armor Plate Detection




Activities
jetbrains-clion
Mar 1 20:45
cpp_pubsub – Detector.cpp
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
TR_Communication_Test › src › cpp_pubsub › src › AutoAim › Detector › Detector.cpp
camera_tester | Debug
Detector.cpp
long Detector::Detect(Mat &input) {
//cout << "111111" << endl;
Mat raw, blurred, gray, binary, dilated, bgr[3], contour;
raw = input.clone();
GaussianBlur(src: raw, dst: blurred, ksize: Size(width: blur_size, height: blur_size), sigmaX: 0);
split(src: raw, mvbegin: bgr);
vector<vector<Point>> contours;
vector<Vec4i> hierarchy;
#ifdef BLUE
gray = bgr[0] - bgr[2];
#else
...
#endif
threshold(src: gray, dst: binary, thresh: 127, maxval: 255, type: THRESH_BINARY);
morphologyEx(src: binary , dst: dilated, op: MORPH_CLOSE, kernel: getStructuringElement(shape: MORPH_RECT, ksize: Size(width: morph_size, he
findContours(image: dilated, contours, hierarchy, mode: RETR_TREE, method: CHAIN_APPROX_SIMPLE, offset: Point(x: 0, y: 0));
detector_debug (raw, & gray, binary, dilated, show_raw: false, show_thresh: true, show_binary: true, show_dilated: false);
cv::Point2f offset= { x: 0, y: 0};
vector<LED_bar> LEDs;
for (int i = 0; i < contours.size(); i++){
LED_bar led = LED_bar(rect: minAreaRect(points: contours[i]));
#ifdef DEBUG
cout << "Led angle: " << led.angle << std::endl;
#endif
LEDs.push_back(led);
}
#ifdef DEBUG
for (int i = 0; i < LEDs.size(); i++) {
Point2f vertices[4];
LEDs[i].rotatedRect.points(pts: vertices);
for (int i = 0; i < 4; i++)
line(img: raw, pt1: vertices[i], pt2: vertices[(i + 1) % 4], color: Scalar(v0: 0, v1: 255, v2: 0), thickness: 2);
Rect brect = LEDs[i].rotatedRect.boundingRect();
rectangle(img: raw, rec: brect, color: Scalar(v0: 255, v1: 0, v2: 0), thickness: 2);
}
#endif
//imshow("rectangles", raw);
Detector::Detect
Detector.cpp
vector<Armor> armors;
for (int i = 0; i < LEDs.size(); i++){
//cout << LEDs.size() << endl;
for (int j = i + 1; j < LEDs.size(); j++) {
Armor armor = Armor(left: LEDs[i], right: LEDs[j]);
#ifdef DEBUG
cout << LEDs.size() << endl;
#endif
if(armor.match_LEDs(img: gray)) {
armors.push_back(armor);
//std::cout << "Left Bar: " << armor.left.angle << std::endl;
//std::cout << "Right Bar: " << armor.right.angle << std::endl;
#ifdef DEBUG
armor.draw_Armor(& raw, roi_offset: offset, drawCenter: false);
#endif DEBUG
}
}
}
#ifdef DEBUG
imshow(winname: "Armors", mat: raw);
#endif
double closest_to_center = DBL_MAX;
double distance_to_center = 0;
Armor final_armor;
for(Armor valid_armor: armors){
distance_to_center = sqrt(x: pow(x: 320 - valid_armor.center.x, y: 2) + pow(x: 240 - valid_armor.center.y, y: 2));
if (distance_to_center < closest_to_center) {
closest_to_center = distance_to_center;
final_armor = valid_armor;
}
}
Point2i final_center = final_armor.center;
//final_armor.draw_Armor(raw, offset, true);
//imshow("Target", raw);
return 0;
}
void Detector::detector_debug (Mat &raw, Mat &gray, Mat &binary, Mat &dilated, bool show_raw , bool show_gray,
bool show_binary, bool show_dilated)
{
if(show_raw) imshow(winname: "raw", mat: raw);
if(show_gray) imshow(winname: "gray", mat: gray);
if(show_binary) imshow(winname: "binary", mat: binary);
Detector::Detect
Version Control
TODO
Problems
Terminal
CMake
Python Packages
Services
Event Log
<no default server> 129:26 LF UTF-8 4 spaces C++: camera_tester | Debug

Armor Plate Detection Test



Supercapacitor

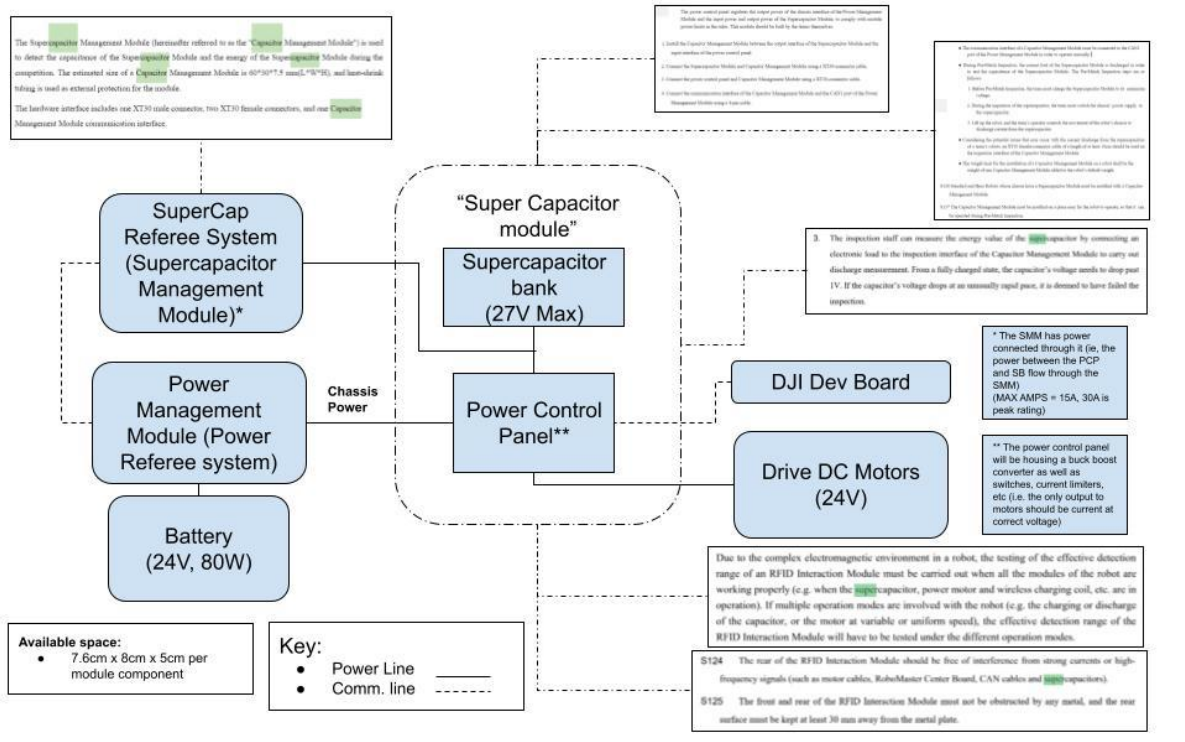
Supercapacitor Functionality

- Allow our ground robots to temporarily bypass the 80W drive motor power limit to achieve controlled bursts of speed as game situations demand it.
- Charge capacitor banks using excess power from the Power Management Module(PMM)
- Allow operators to monitor capacitor charge states using HUD displays

Relevant Modules:

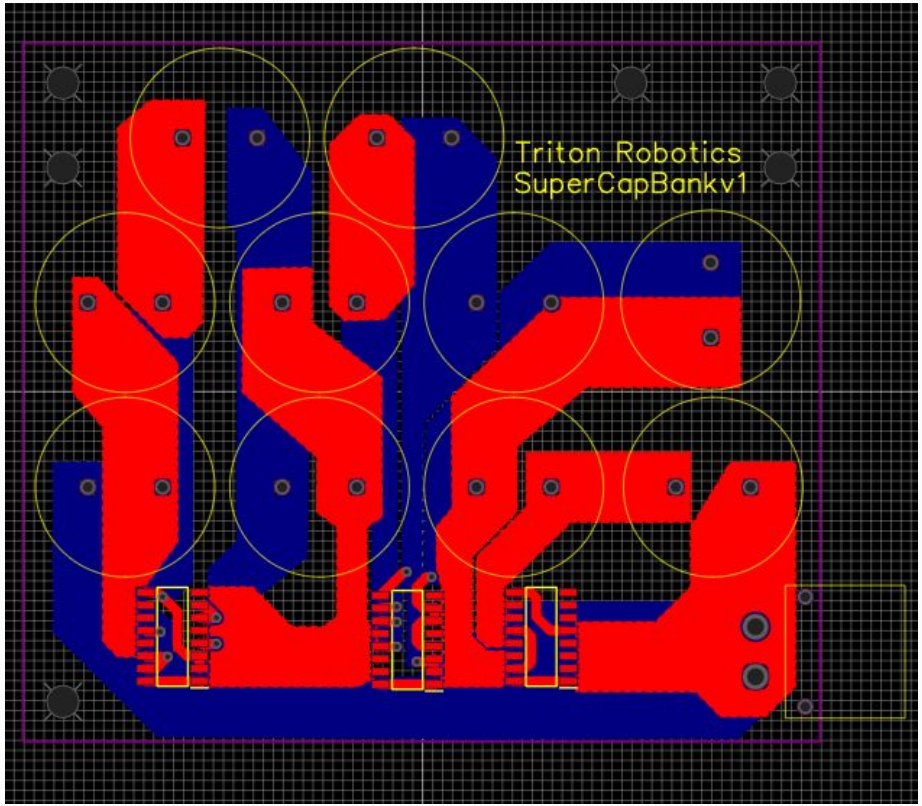
- Power Control Board
- Supercapacitor Bank

Supercapacitor Power System



- Generalized wiring diagram

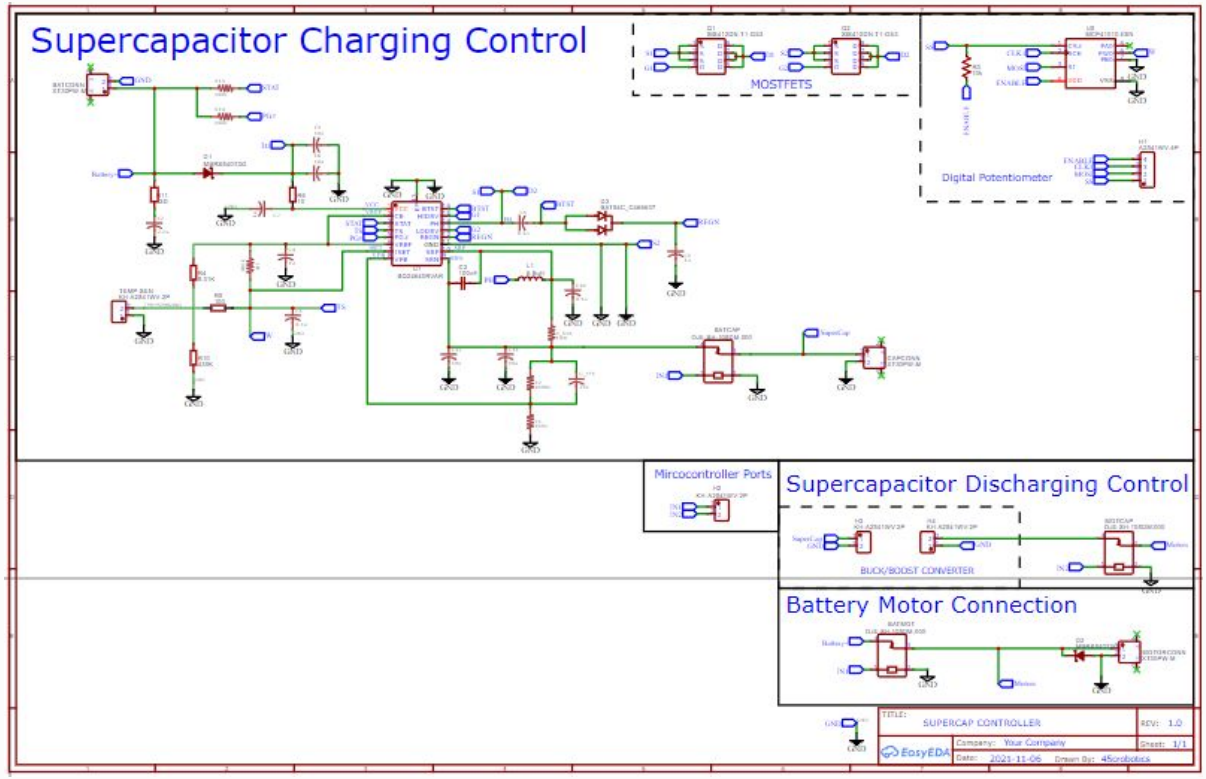
Supercapacitor Power System



Super Capacitor bank board file

- Includes passive charge balancing

Supercapacitor Power System



- Control Board Schematic
- Features:
  - Capacitor current control
  - Temperature sensing/lockout
- Utilizes pre packaged modules:
  - Boost converter
  - Power Management IC