



Midterm Presentation

Triton Robotics
University of California, San Diego

Table of Contents

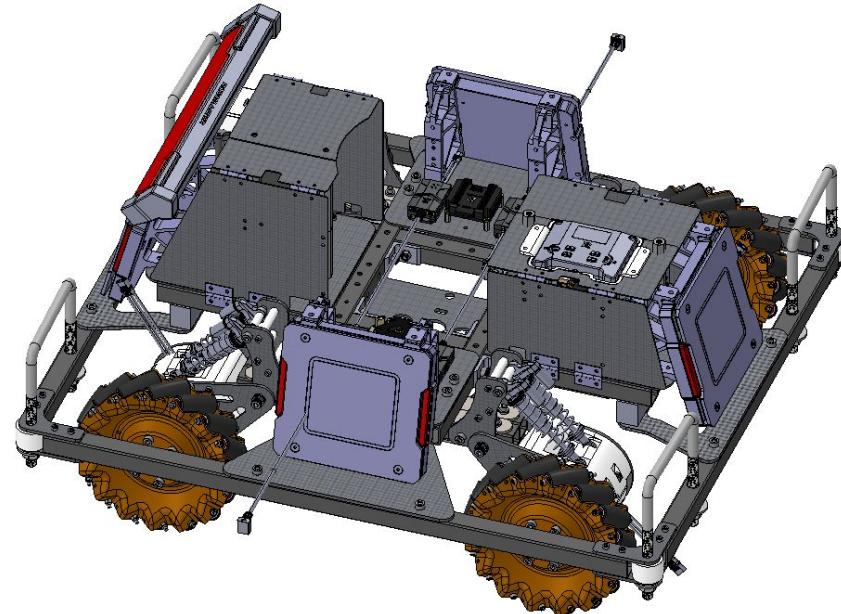
1. [Infantry](#)
2. [Hero](#)
3. [Sentry](#)
4. [Engineer](#)
5. [Drone](#)
6. [Dart](#)
7. [Embedded](#)
8. [Computer Vision](#)
9. [SuperCapacitor](#)

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

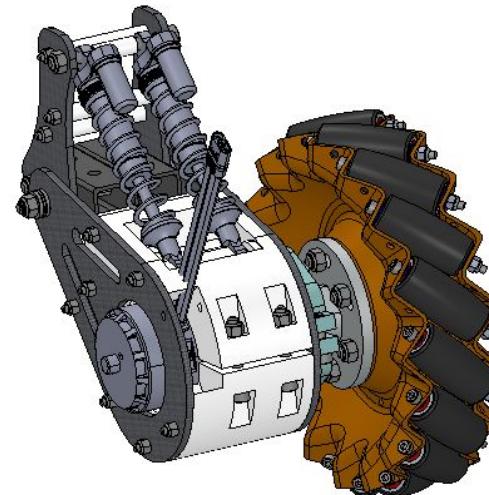
- **Suspension**

- **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

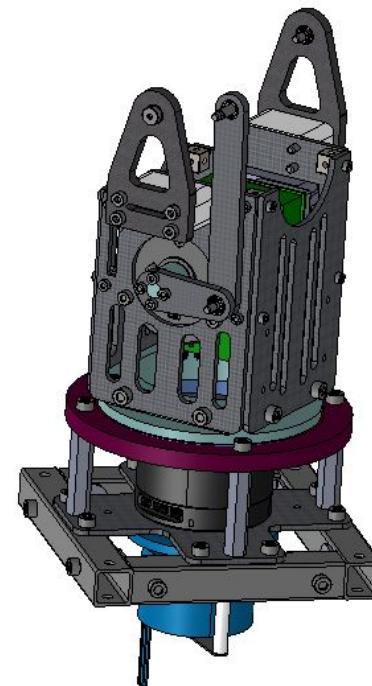
- **Turret Base**

- **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:**

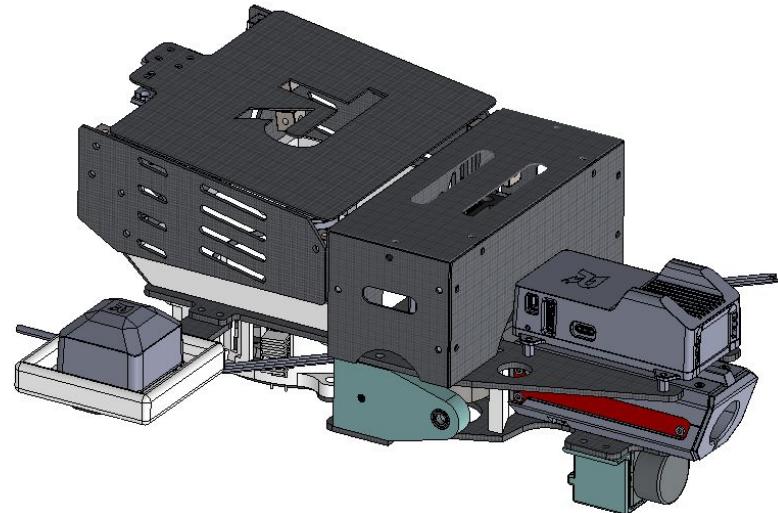
- Reduce number of fasteners (trying to include tapped holes in machined parts)



CAD Design - Turret

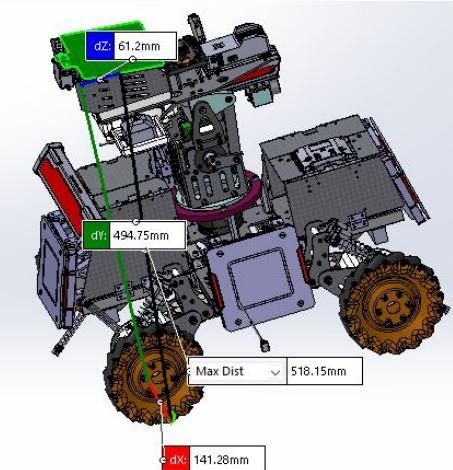
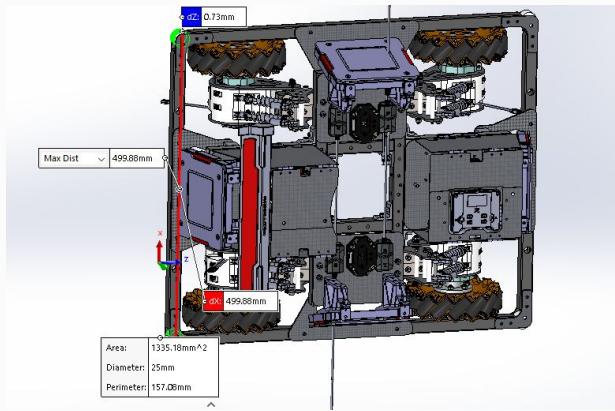
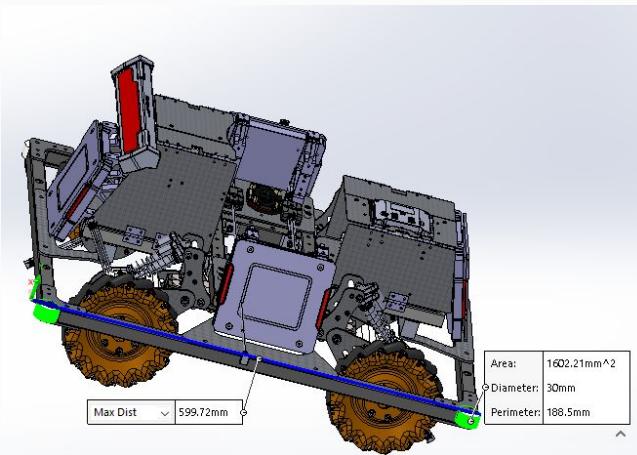
- **Turret**

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

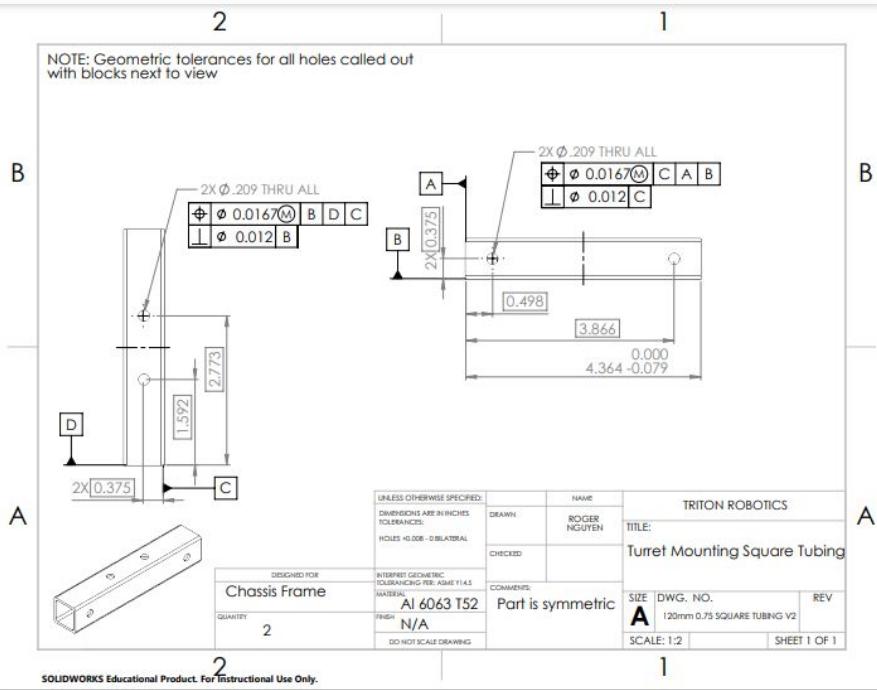


Dimensions

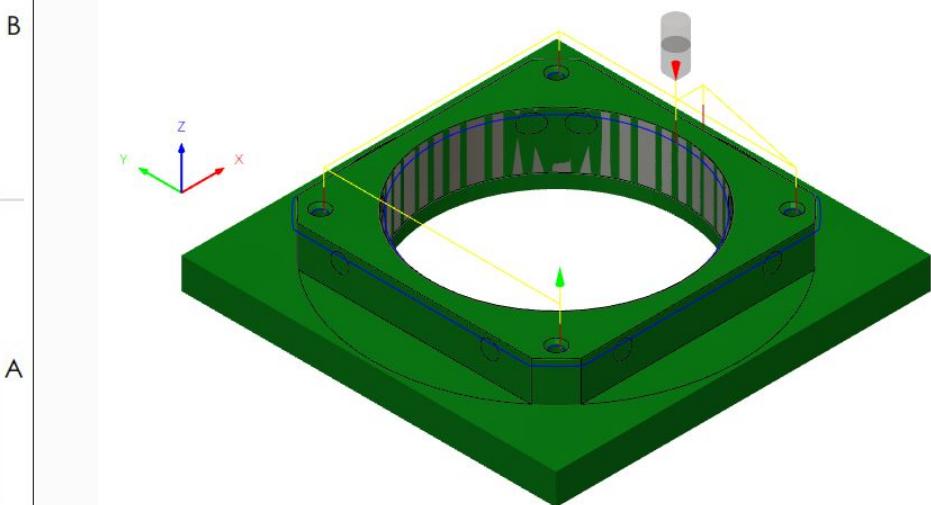
- (L,W,H): (599.72, 499.88, 518.15) mm



Manufacturing Showcase

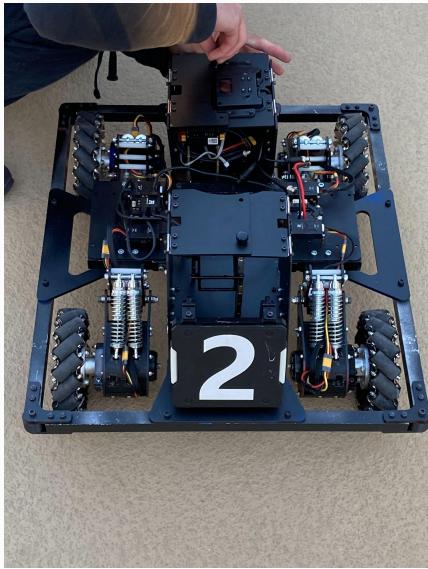


Drawing for Milling

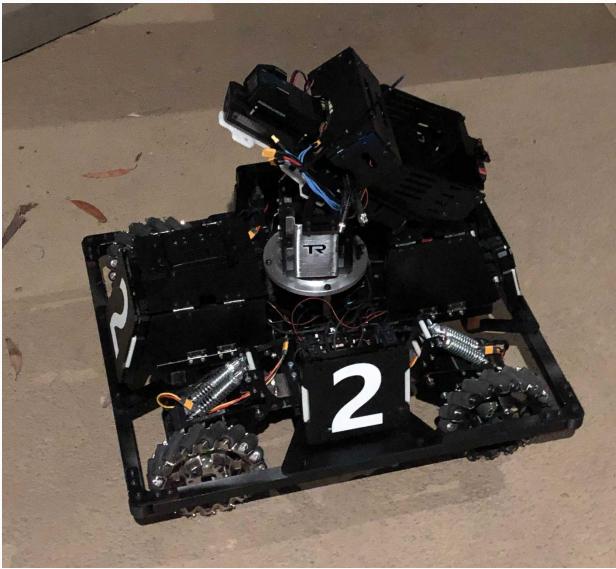


Program for CNC Milling

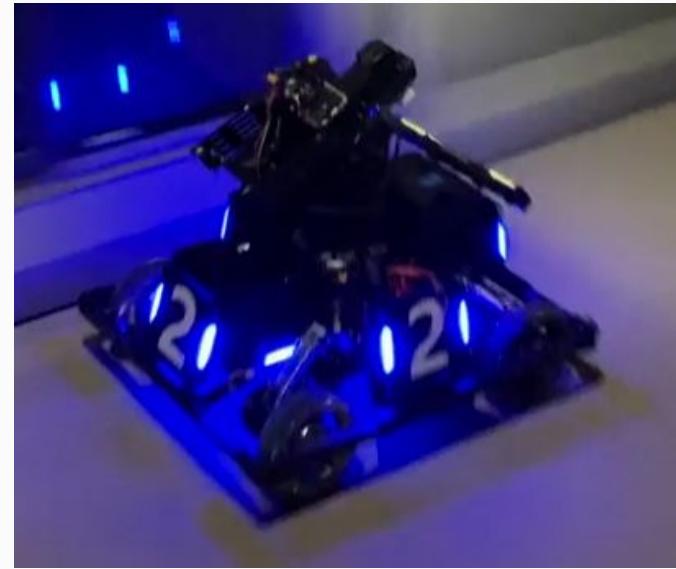
Physical Model Showcase



Robot without Turret

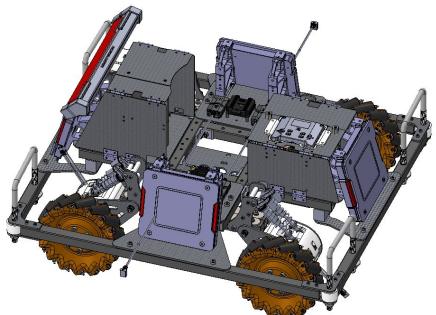


Full Robot

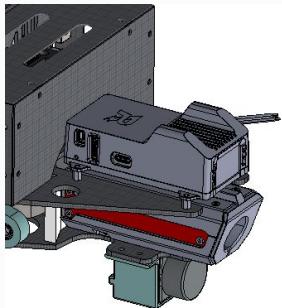


Full Robot Online

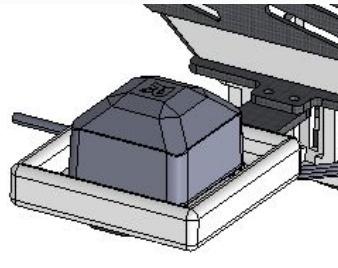
Referee System Showcase



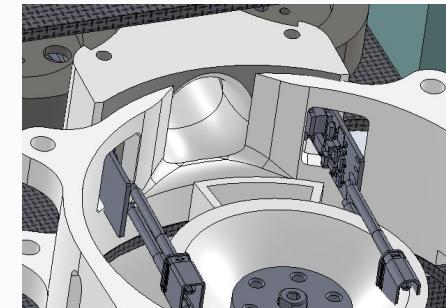
4 armor plates + 1 HP Bar



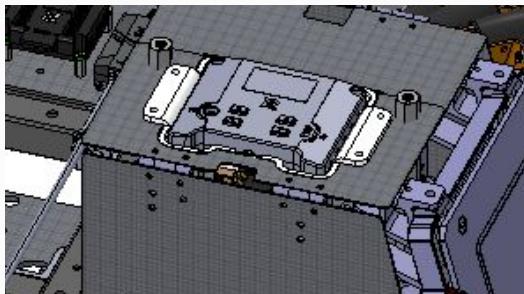
17mm Speed Monitor + VTM



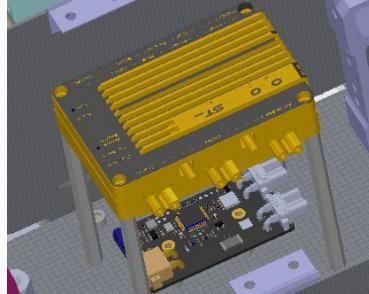
GPS



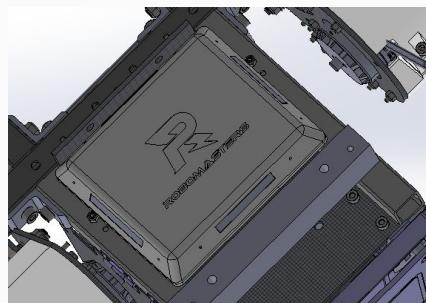
Fluorescent LED Driver



Main Control Module

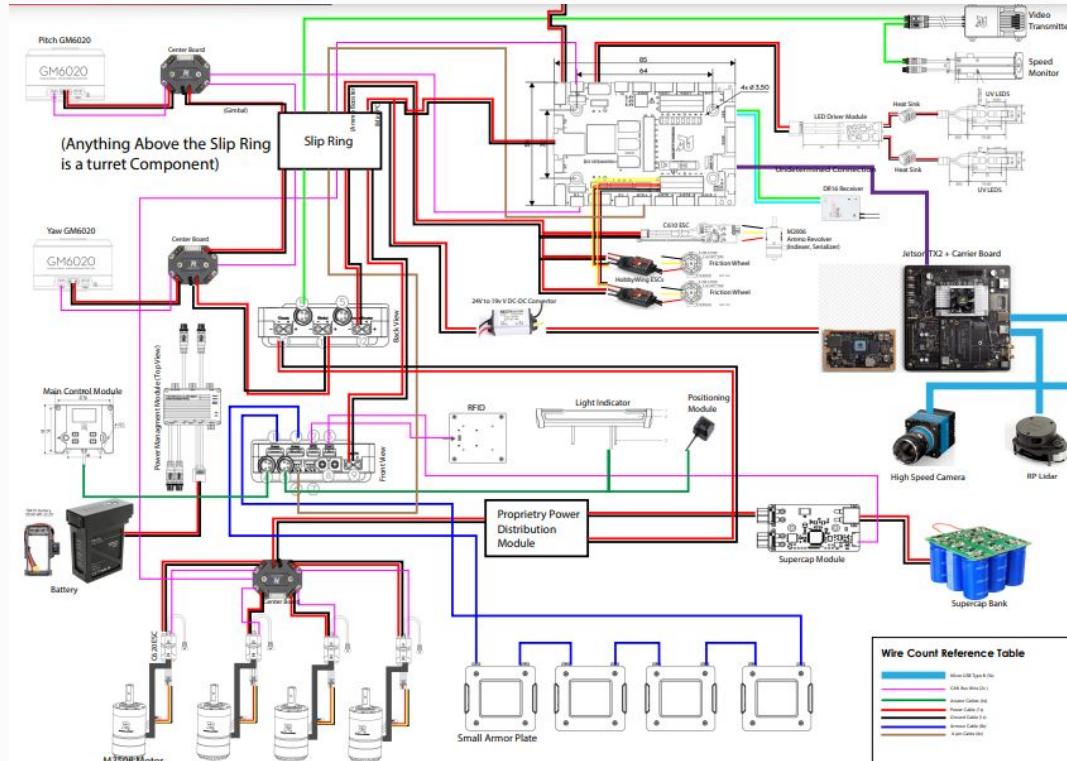


PMM + SuperCap Module



RFID

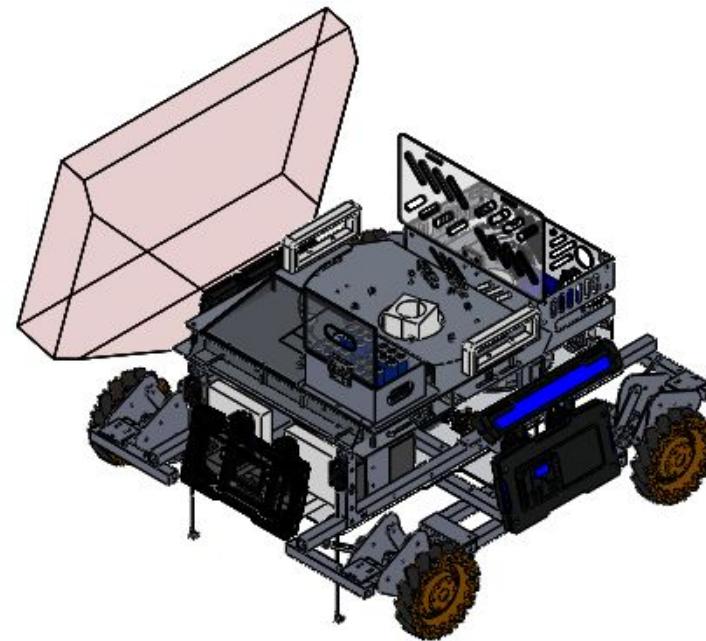
Wiring Diagram Showcase



Hero

CAD Design - Chassis

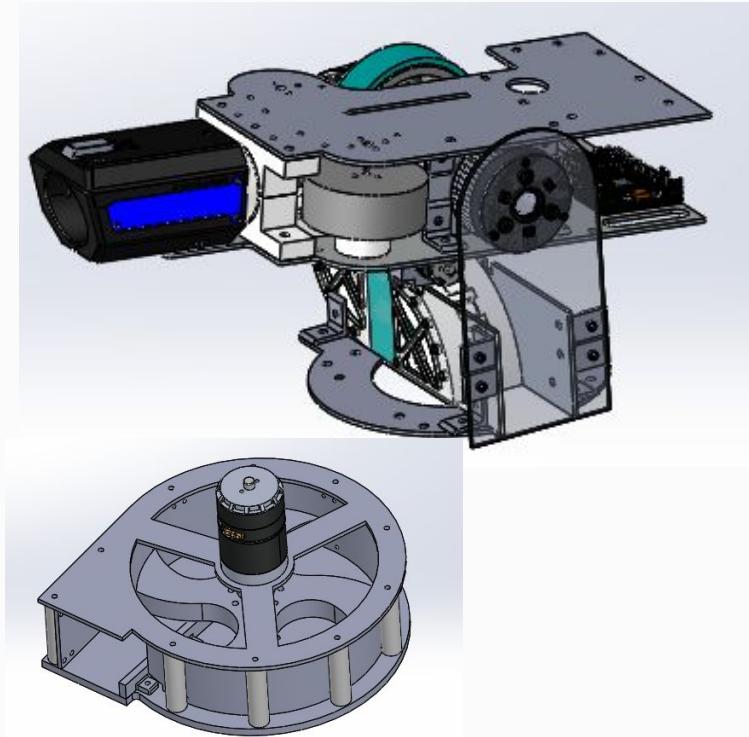
- **Chassis**
 - **Capability:**
 - Easy access to internals
 - Experimental suspension design for versatile movement
 - Store 45 42mm rounds
 - **To be improved:**
 - Wiring and electronic mount
 - Reduce size/weight
 - Suspension



CAD Design - Gimbal/ Serializer

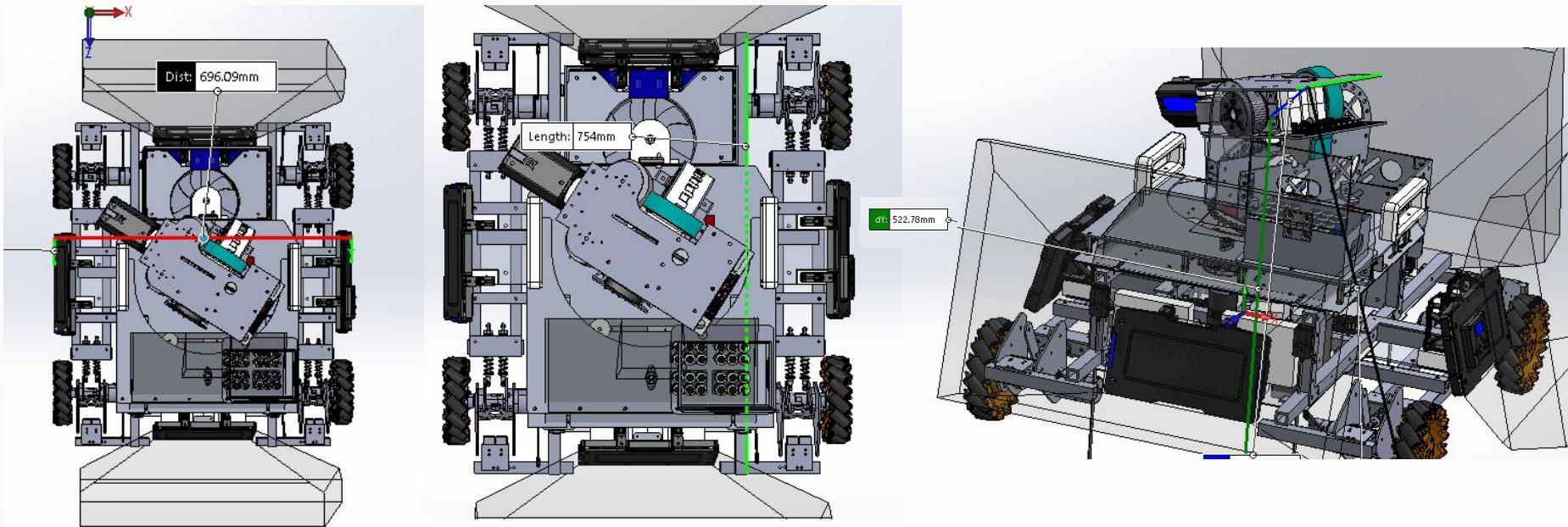
- Gimbal

- 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)

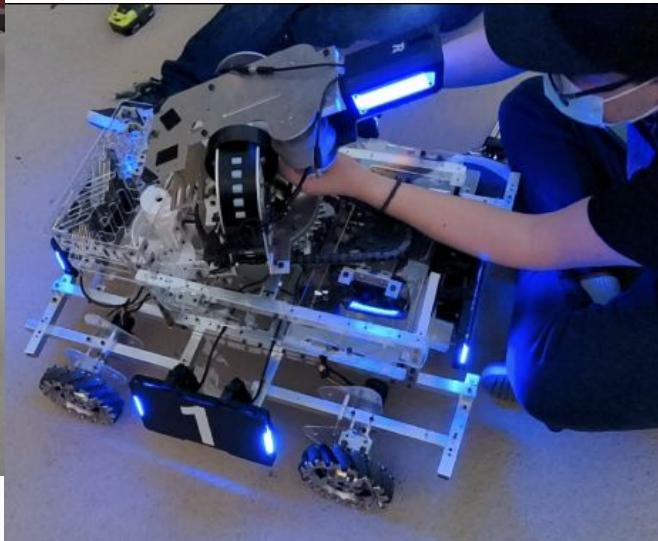
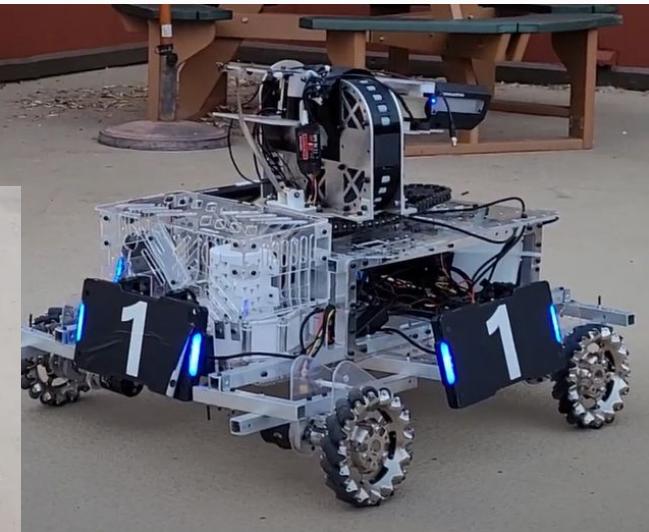
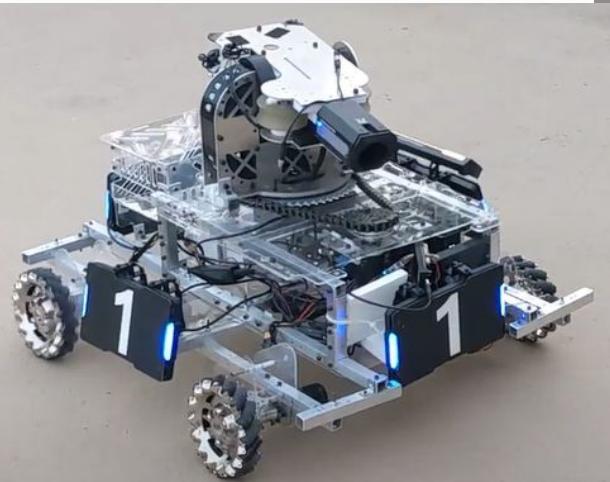


Dimensions

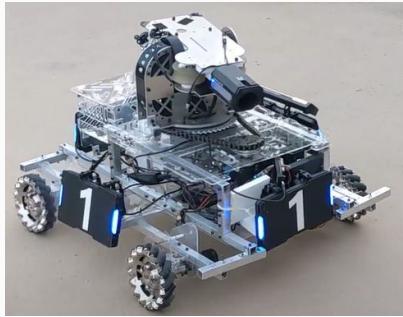
- 754x694.5x522.78 LWH



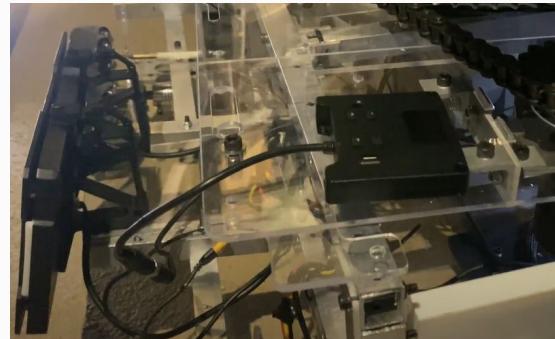
Hero Prototype



Hero Referee System



4 Large Armor
Modules & Light
Indicator



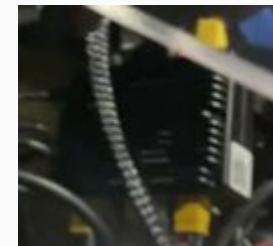
Main Control Module



RFID Module



42mm Speed Monitor
& Transmitter

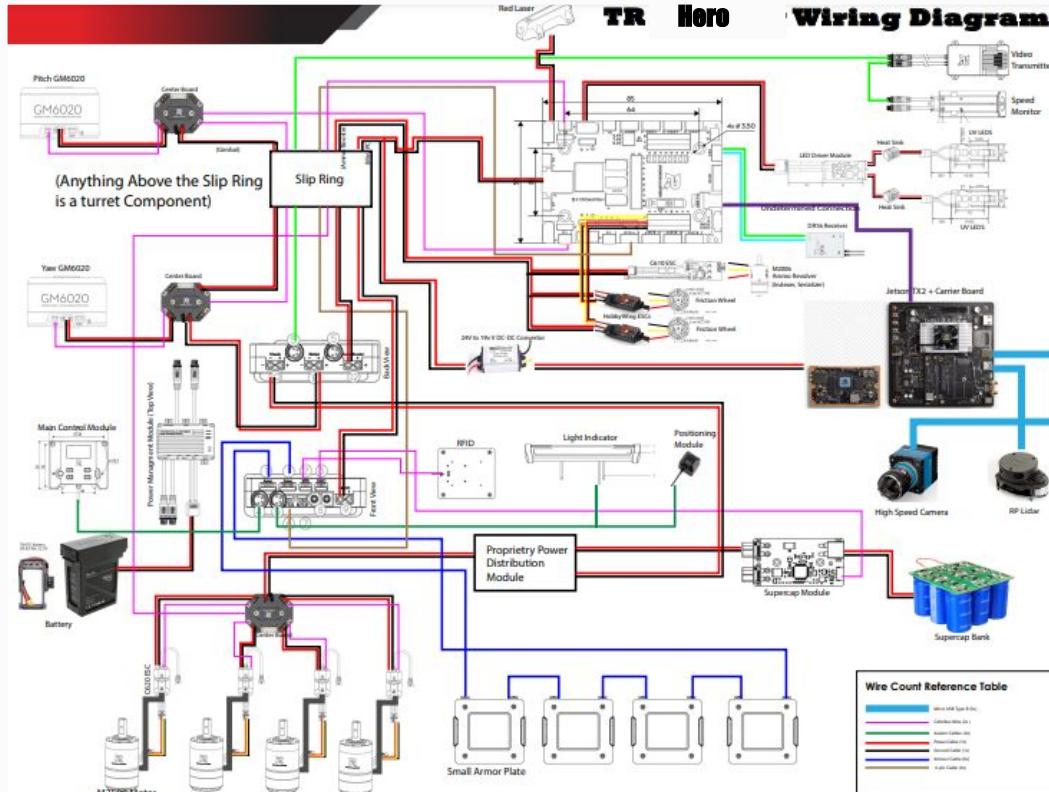


Power Management Module



Positioning
System mount

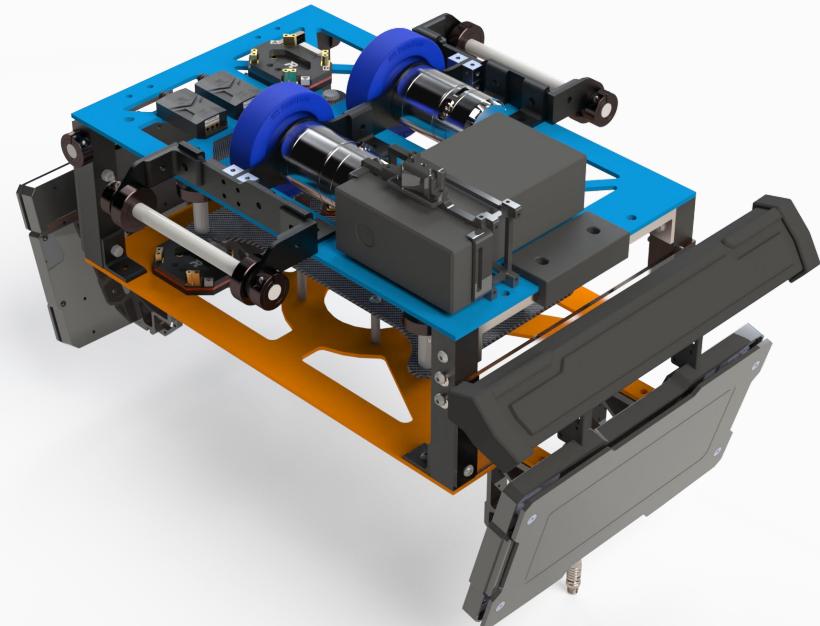
Wiring Diagram



Sentry

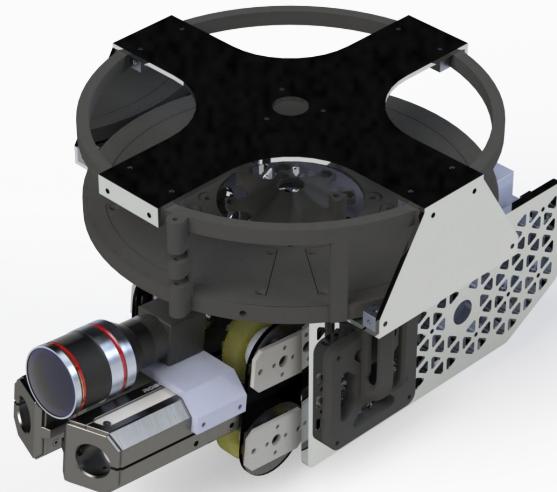
CAD Design -Chassis

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



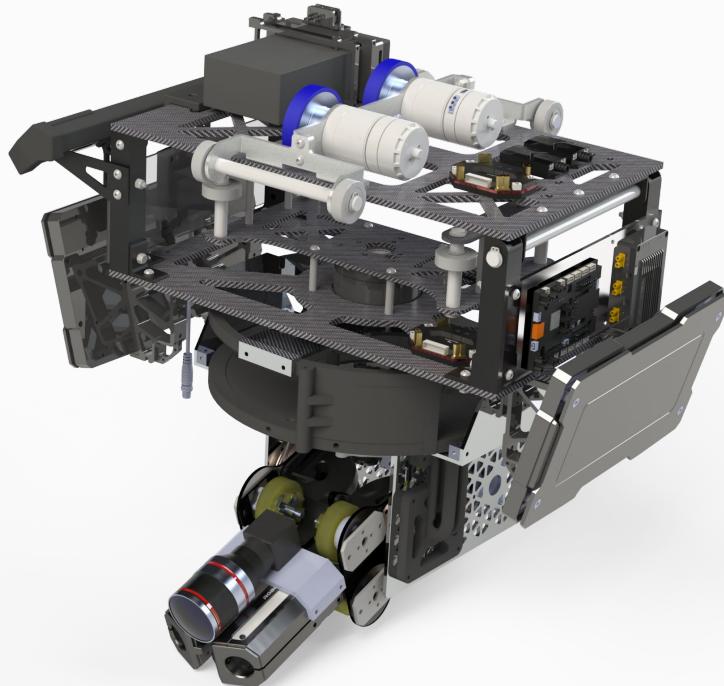
CAD Design -Turret

- **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**
 -



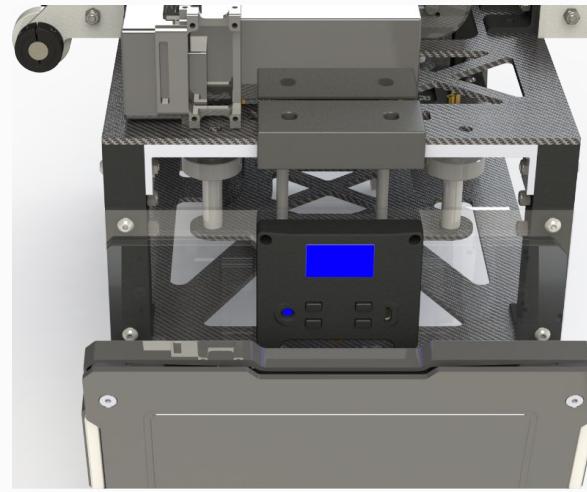
Referee System Showcase



Power Management
Module



17mm Speed Monitor
Module



Main Control Module

Engineer

CAD Design - Chassis

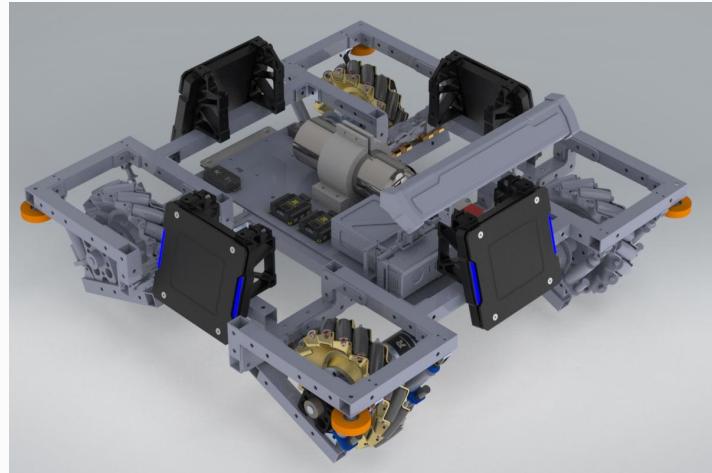
- **Chassis**

- **Capability:**

- **Relatively stable under violent crash**
 - **Easy Access of wiring**
 - **Roller on corners of chassis to prevent stuck in arena corners**

- **To be improved:**

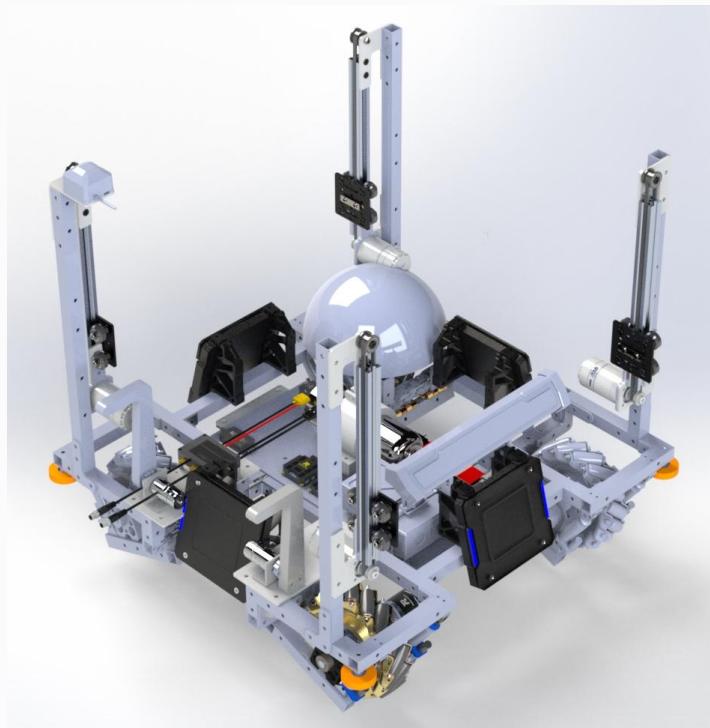
- **Side bars of suspension module are too low that may get stuck on bumpy road.**



CAD Design - Lifting Mechanism

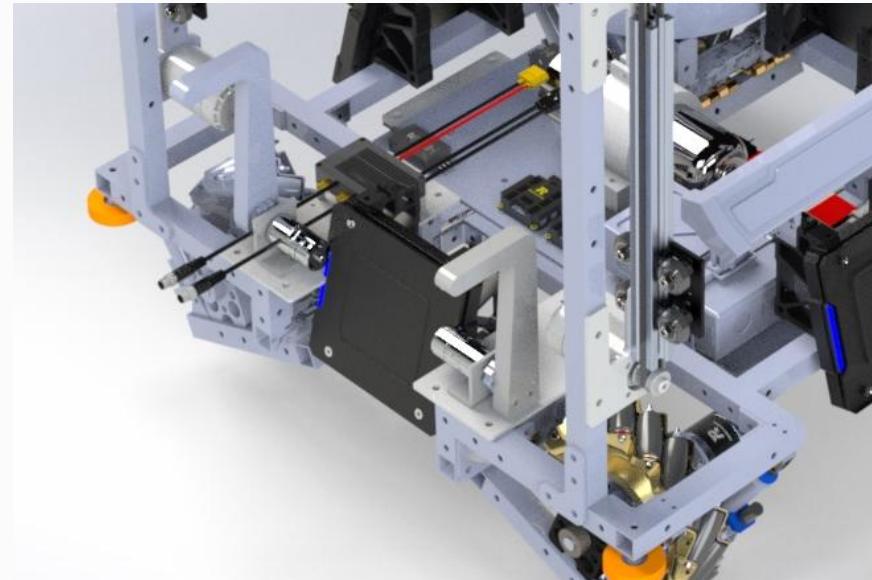
- **Lifting Mechanism**

- **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**



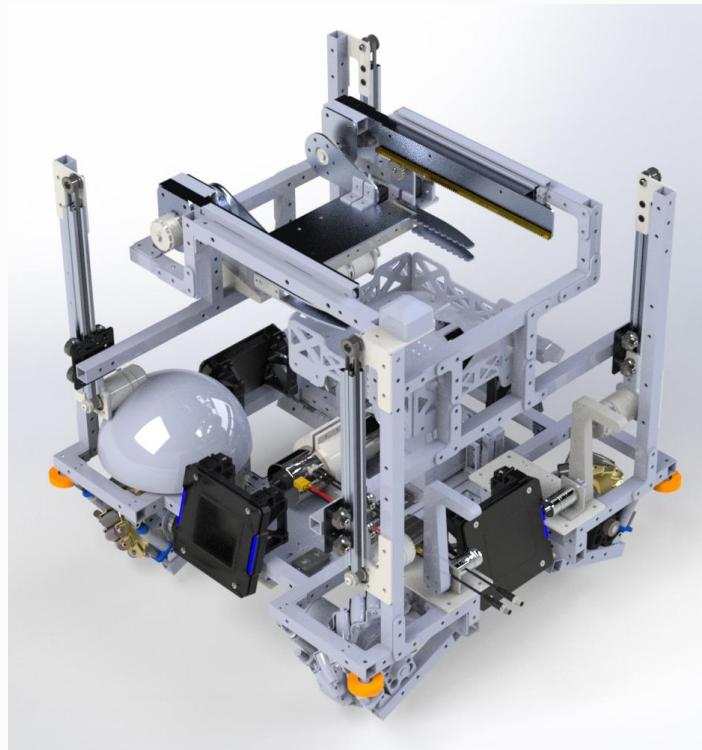
CAD Design - Rescue Mechanism (not tested)

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

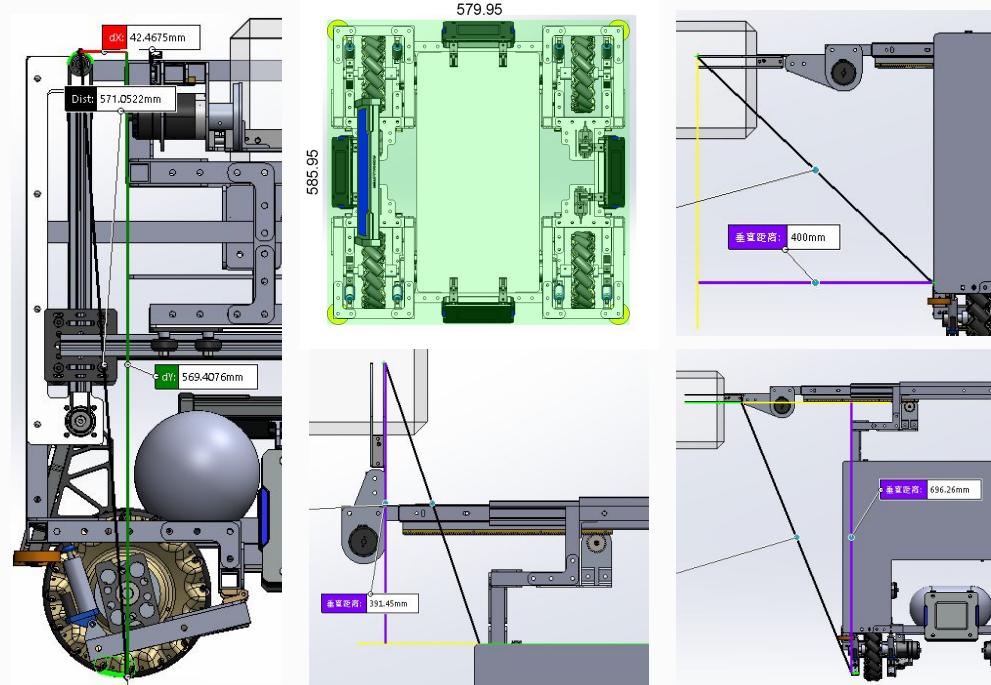
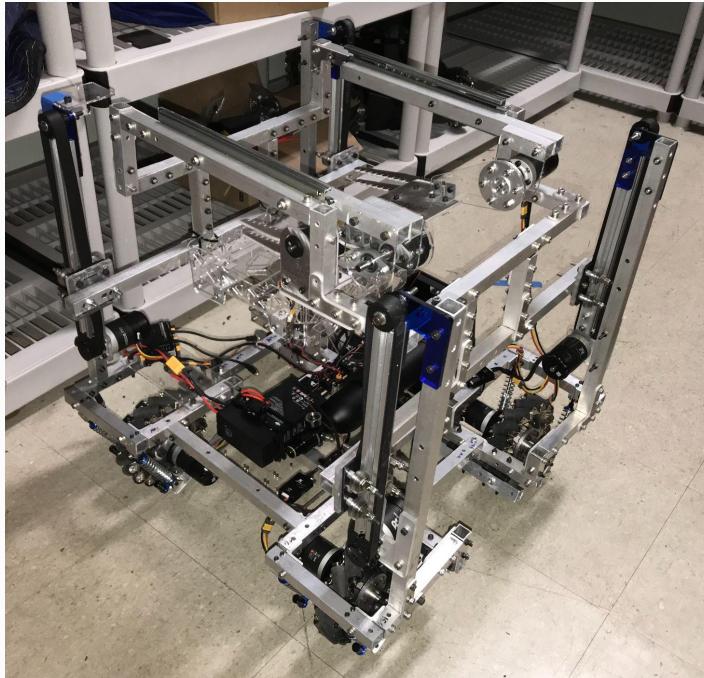


CAD Design - Claw and Mineral Box

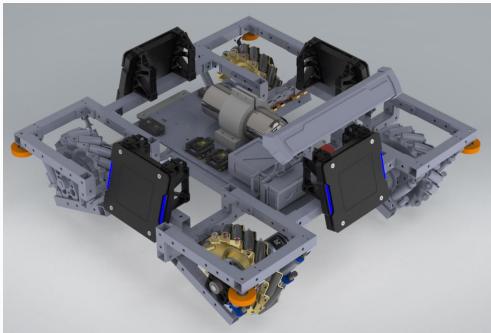
- **Claw and Mineral Box**
 - **Capability:**
 - Extend 400mm
 - Store one mineral in the rear
 - **To be improved:**
 - Adding space between the sides of the claw to have more tolerance for flipping back
 - Close the grippers tighter



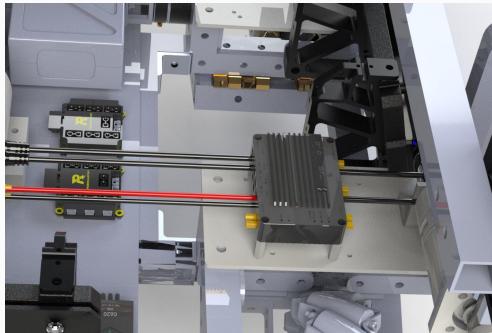
Engineer Prototype



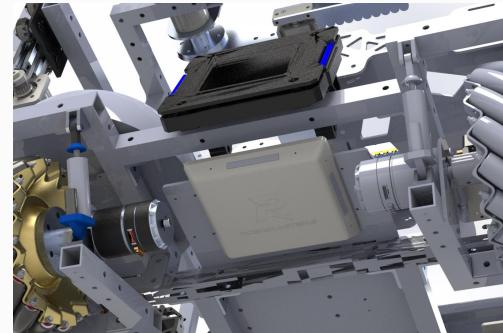
Referee System Showcase



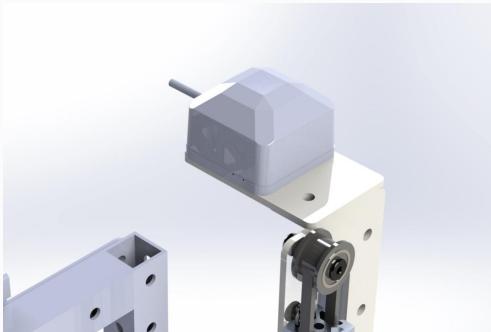
4 armor plates + 1 HP Bar



Power Management Module



RFID



GPS

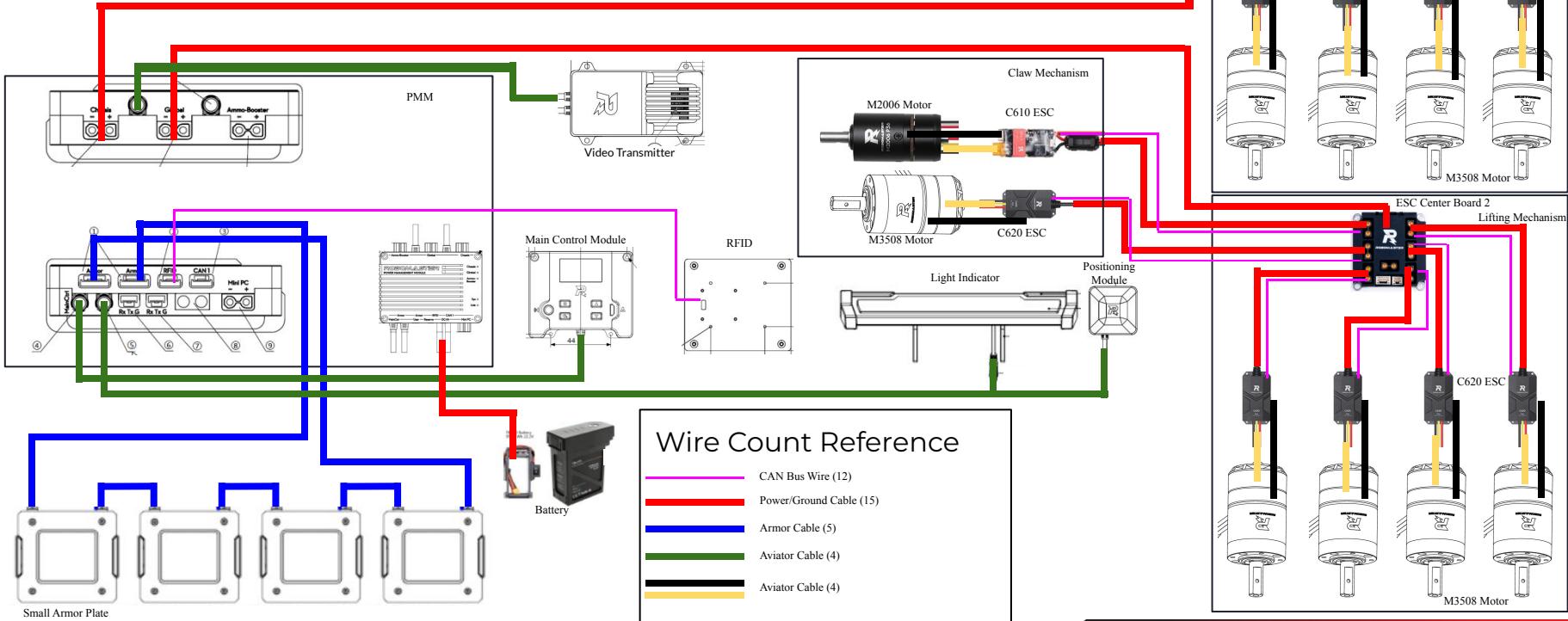


VTM



Main Control Module

Engineer Wiring Diagram



Drone

CAD Design - Turret

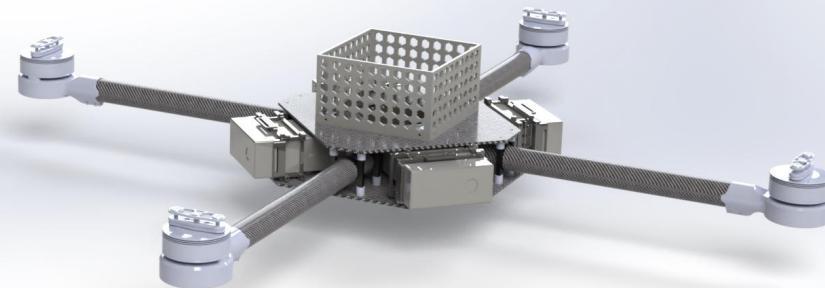
- **Turret**
 - **Capability:**
 - Shoot 17mm balls at 25 m/s
 - Allows for easy dev board access
 - **To be improved:**
 - Not a lot of space to mount the mini PC
 - Needs better wire management



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**

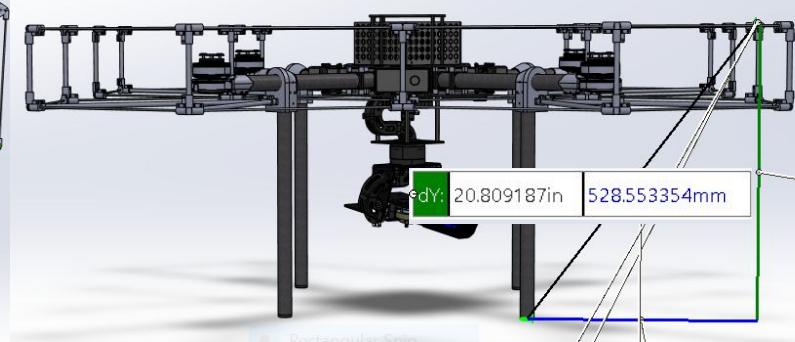
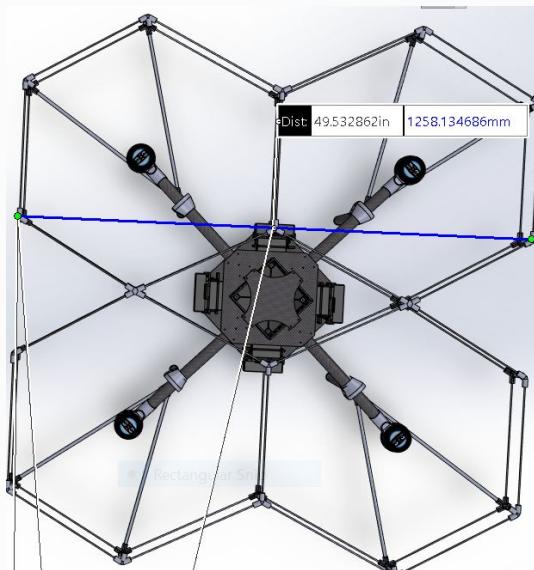
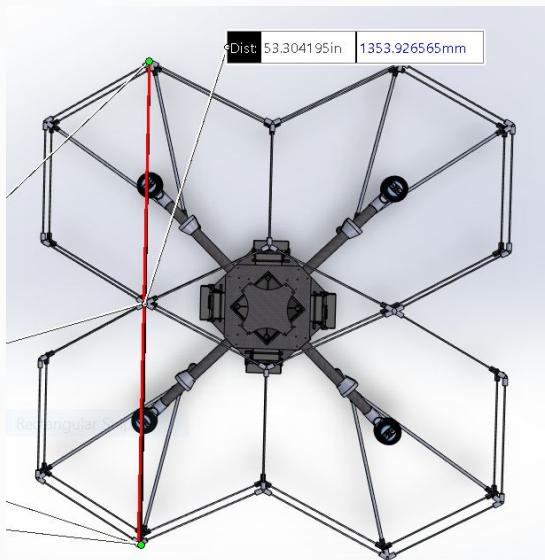
- **To be improved:**

- **Landing gear legs are not stable**
 - **Cage needs more secure mounting to the airframe**



Dimensions

- 1353x1258x529mm LWH



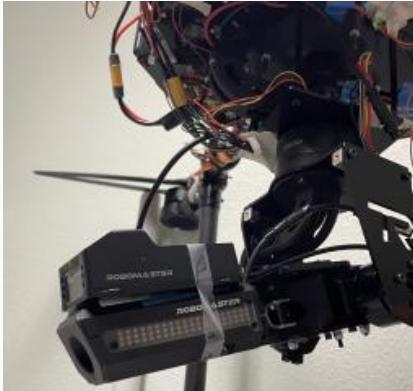
Drone Prototype



Referee System Showcase



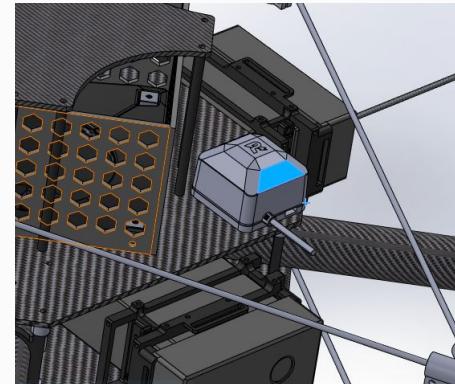
Power Management
Module



17mm Speed Monitor
Module and VTM



Main Control Module

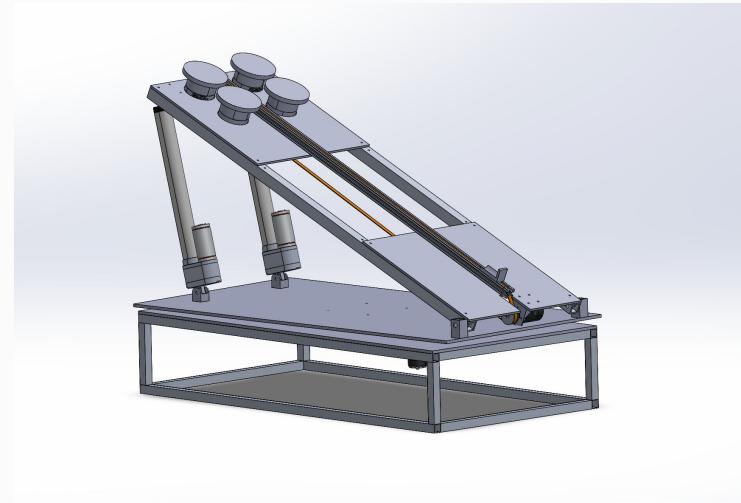


GPS

Dart

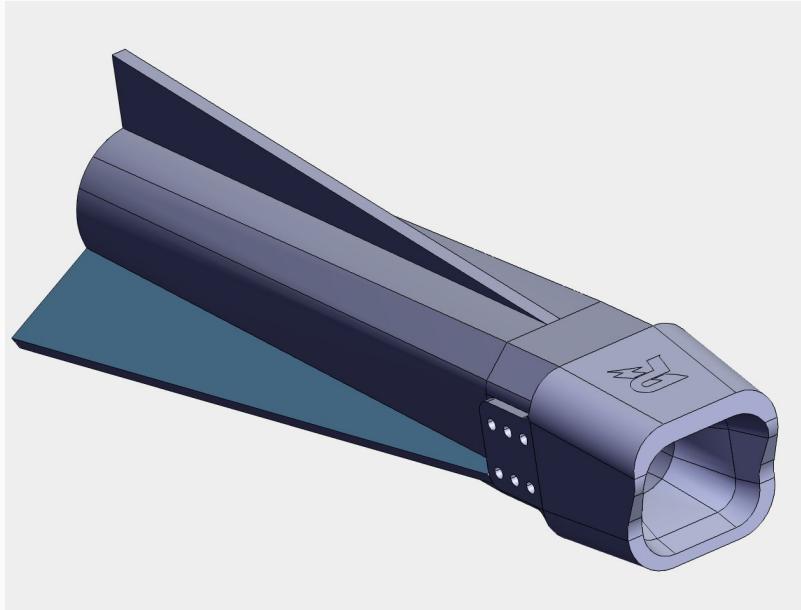
CAD Design - Launcher

- **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

```
15 void Remote::read()
16 {
17     // Read next byte if available and more needed for the current packet
18     //printf("Attempting to read \n");
19     receiver.read(rxBuffer, sizeof(rxBuffer));
20
21     for (int i = 0; i < sizeof(rxBuffer); i++) {
22         rxBuffer[i] = ~rxBuffer[i];
23         //printf("%dt", rxBuffer[i]);
24     }
25     //printf("\n");
26     //printf("Parsing buffer\n");
27     parseBuffer();
28     clearRxBuffer();
29 }
30
31 bool Remote::isConnected() const { return connected; }
32
33 float Remote::getChannel(Channel ch) const
34 {
35     switch (ch)
36     {
37         case Channel::RIGHT_HORIZONTAL:
38             return remote.rightHorizontal / STICK_MAX_VALUE;
39         case Channel::RIGHT_VERTICAL:
40             return remote.rightVertical / STICK_MAX_VALUE;
41         case Channel::LEFT_HORIZONTAL:
42             return remote.leftHorizontal / STICK_MAX_VALUE;
43         case Channel::LEFT_VERTICAL:
44             return remote.leftVertical / STICK_MAX_VALUE;
45     }
46     return 0;
47 }
48
49 Remote::SwitchState Remote::getSwitch(Switch sw) const
50 {
51     switch (sw)
52     {
53         case Switch::LEFT_SWITCH:
54             return remote.leftSwitch;
55         case Switch::RIGHT_SWITCH:
56             return remote.rightSwitch;
57     }
58     return SwitchState::UNKNOWN;
59 }
```



```
130     {
131         return;
132     }
133
134     // mouse input
135     remote.mouse.x = rxBuffer[6] | (rxBuffer[7] << 8); // x axis
136     remote.mouse.y = rxBuffer[8] | (rxBuffer[9] << 8); // y axis
137     remote.mouse.z = rxBuffer[10] | (rxBuffer[11] << 8); // z axis
138     remote.mouse.l = static_cast<bool>(rxBuffer[12]); // left button click
139     remote.mouse.r = static_cast<bool>(rxBuffer[13]); // right button click
140
141     // keyboard capture
142     remote.key = rxBuffer[14] | rxBuffer[15] << 8;
143     // Remote wheel
144     remote.wheel = (rxBuffer[16] | rxBuffer[17] << 8) - 1024;
145
146     remote.updateCounter++;
147 }
148
149 void Remote::clearRxBuffer()
150 {
151     // Reset bytes read counter
152     currentBufferIndex = 0;
153     // Clear remote rxBuffer
154     for (int i = 0; i < REMOTE_BUF_LEN; i++)
155     {
156         rxBuffer[i] = 0;
157     }
158     // Clear Usart1 rxBuffer
159 }
160
161
162 void Remote::reset()
163 {
164     remote.rightHorizontal = 0;
165     remote.rightVertical = 0;
166     remote.leftHorizontal = 0;
167     remote.leftVertical = 0;
168     remote.leftSwitch = SwitchState::UNKNOWN;
169     remote.rightSwitch = SwitchState::UNKNOWN;
170     remote.mouse.x = 0;
171     remote.mouse.y = 0;
172     remote.mouse.z = 0;
173     remote.mouse.l = 0;
174     remote.mouse.r = 0;
175     remote.key = 0;
```

General - Remote Control

```
BUFSIZE serial 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 priori
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.
void parseBuffer();

```

```
7  */
8  class Remote
9  {
10 public:
11     Remote(PinName tx, PinName rx);
12
13 /**
14  * Specifies a particular joystick.
15  */
16 enum class Channel
17 {
18     RIGHT_HORIZONTAL,
19     RIGHT_VERTICAL,
20     LEFT_HORIZONTAL,
21     LEFT_VERTICAL
22 };
23
24 /**
25  * Specifies a particular switch.
26  */
27 enum class Switch
28 {
29     LEFT_SWITCH,
30     RIGHT_SWITCH
31 };
32
33 /**
34  * Different switch orientations.
35  */
36 enum class SwitchState
37 {
38     UNKNOWN,
39     UP,
40     MID,
41     DOWN
42 };
43
44 /**
45  * A list of the particular keys to interact with, in bit order.
46  */
47 enum class Key
48 {
49     W = 0,
50     S,
51     A,

```

General - Servo

```
#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;
}
```

```
#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 Z: // receive from Keyboard
default: // when remote not connected
    break;

}

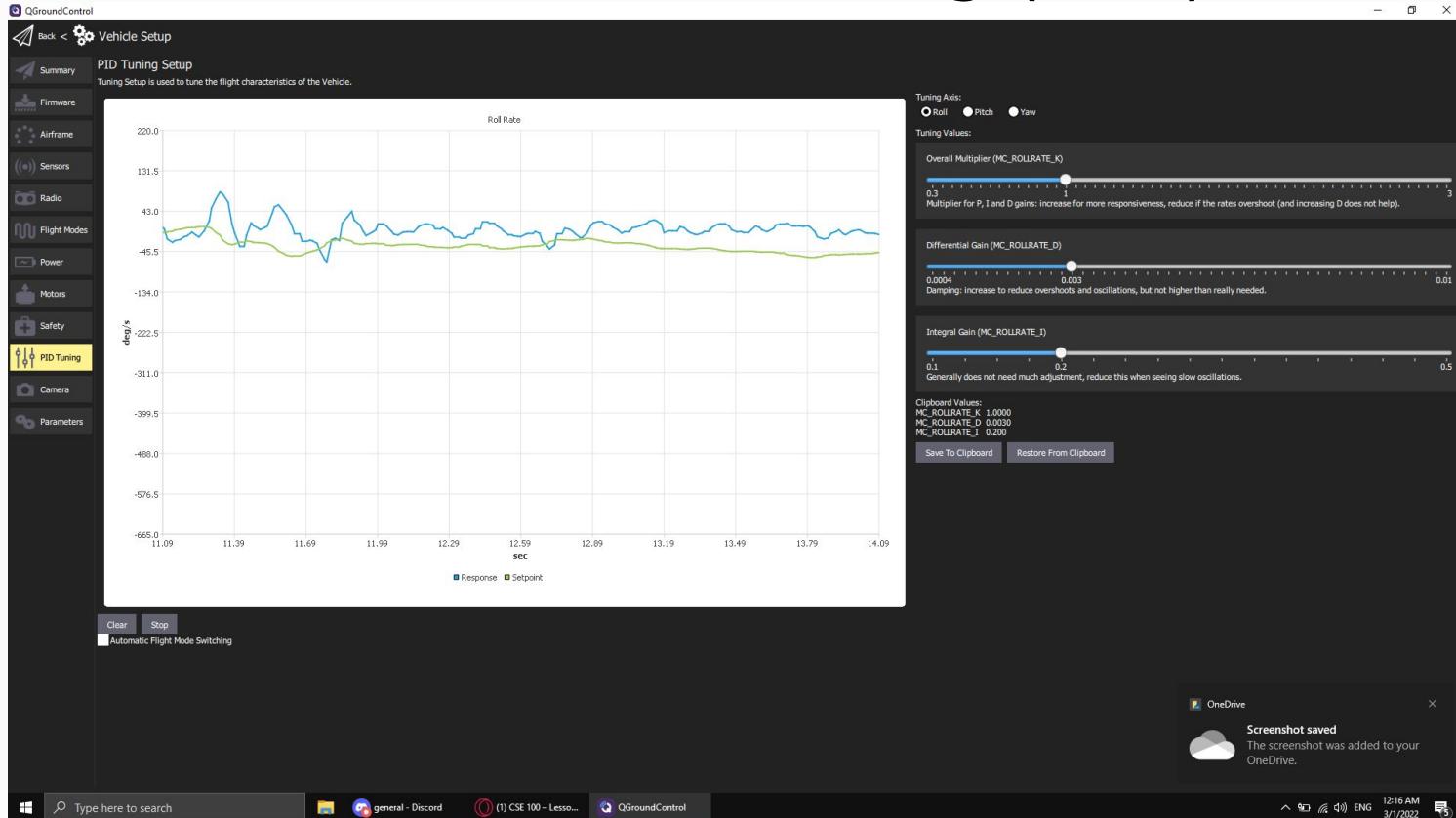
sig->gimbal.pitch_ecd += (short)((joyRightY * JOY_TO_ECD) * 0.05f); // pitch don't have the same
float raw_x = joyLeftX * JOY_TO_RPM * DRIVE_SPEED_DEFAULT;
float raw_y = joyLeftY * JOY_TO_RPM * DRIVE_SPEED_DEFAULT;

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

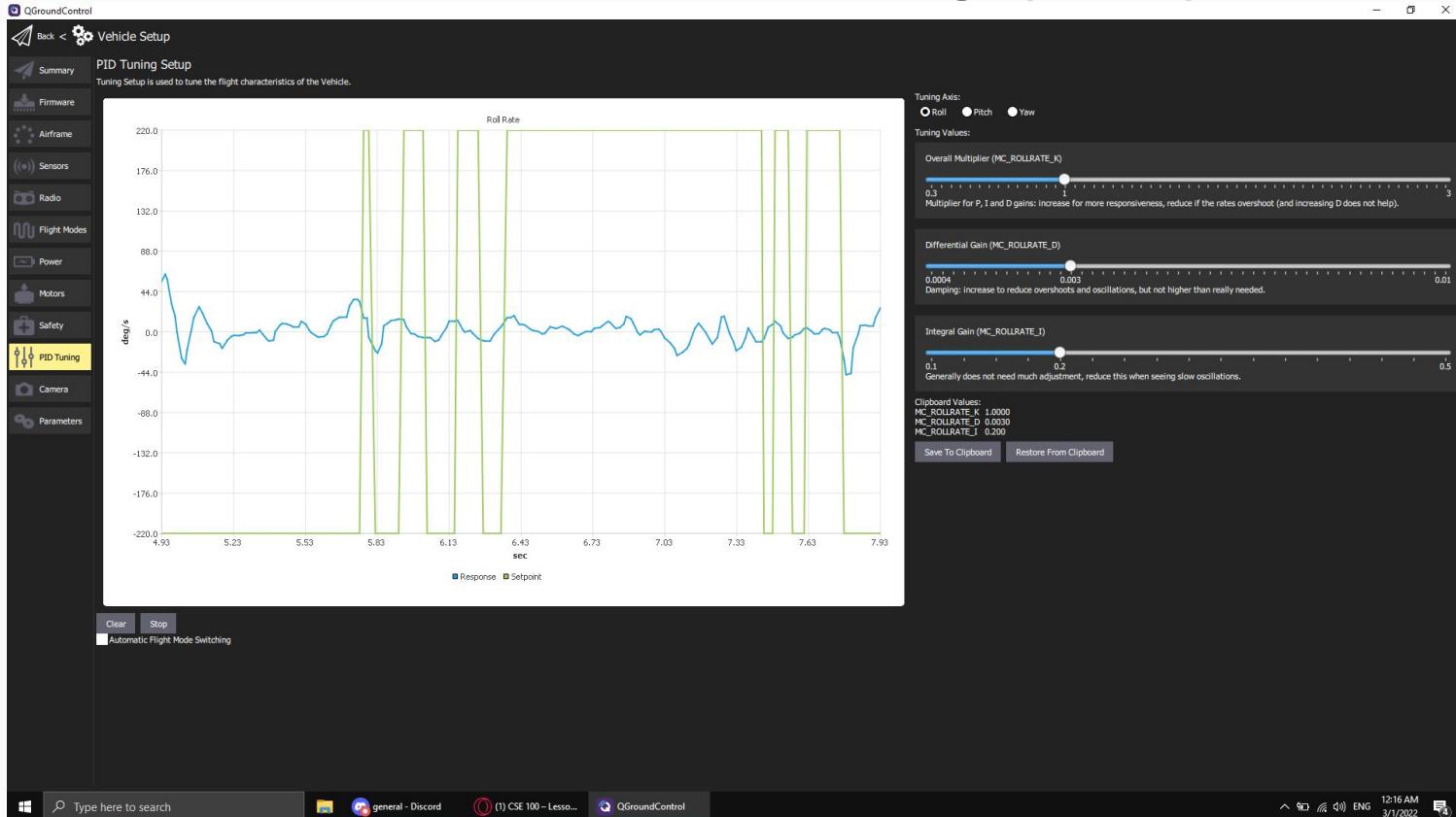
// apply rotational matrix
x = (float) (raw_x * cos(radian) - raw_y * sin(radian));
y = (float) (raw_x * sin(radian) + raw_y * cos(radian));

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);
```

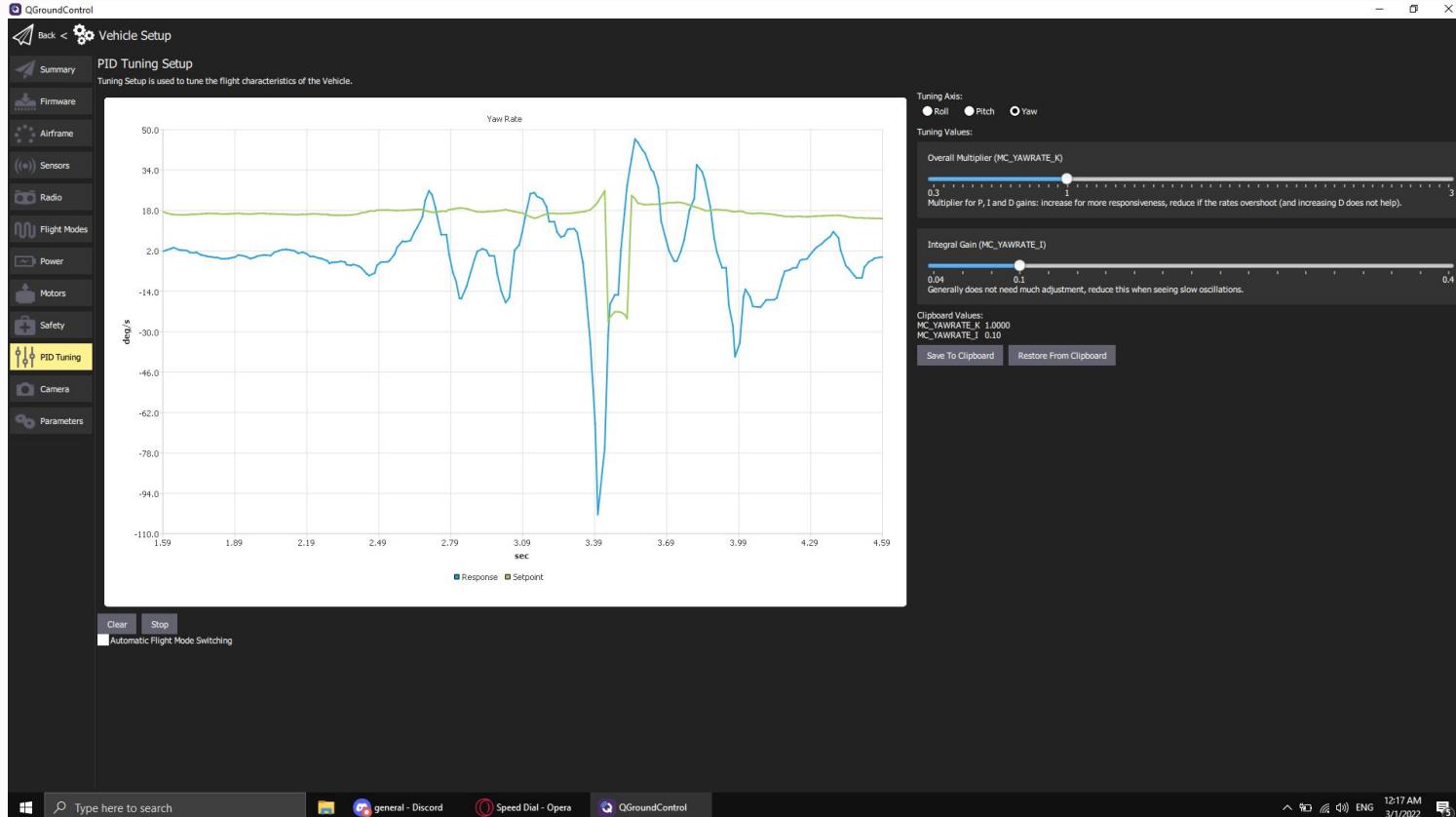
Drone - PID Tuning (Roll)



Drone - PID Tuning (Roll)



Drone - PID Tuning (Yaw)



Drone - Sensor

QGroundControl

Back < Vehicle Setup

Summary Sensors Setup

Sensors Setup is used to calibrate the sensors within your vehicle.

Firmware Airframe Sensors Gyroscope Accelerometer Level Horizon Cancel

Rotates the vehicle continuously as shown in the diagram until marked as Completed.

Rotate the vehicle continuously as shown in the diagram until marked as Completed.

3D view of a white drone with a green circular arrow indicating rotation direction. The status is "Incomplete".

3D view of a white drone from above, showing it has been rotated. The status is "Incomplete".

3D view of a white drone from behind, showing it has been rotated. The status is "Incomplete".

3D view of a white drone from the front, showing it has been rotated. The status is "Incomplete".

3D view of a white drone from the side, showing it has been rotated. The status is "Incomplete".

3D view of a white drone from the other side, showing it has been rotated. The status is "Incomplete".

3D view of a white drone from the back, showing it has been rotated. The status is "Incomplete".

3D view of a white drone from the front, showing it has been rotated. The status is "Incomplete".

3D view of a white drone from the side, showing it has been rotated. The status is "Incomplete".

3D view of a white drone from the other side, showing it has been rotated. The status is "Incomplete".

Type here to search

general - Discord Speed Dial - Opera QGroundControl

12:16 AM ENG 3/1/2022

Computer Vision

Armor Plate Detection

Activities Mar 1 20:43

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help

TR_Communication_Test src cpp_pubsub src AutoAim Armor

Detector.h Detector.cpp Armor

```
#include "Armor.h"
using namespace cv;
Armor::Armor() {}

Armor::Armor(const LED_bar &left, const LED_bar &right) {
    this->left = left;
    this->right = right;
    this->xerror_angle = fabs( (left.rotatedRect.angle - right.rotatedRect.angle) / 2 );

    rect.size.width = 100 * static_cast<int>(left.rotatedRect.center.x - right.rotatedRect.center.x);
    rect.size.height = static_cast<int>((left.rotatedRect.size.height + right.rotatedRect.size.height) / 2);
    center.x = static_cast<int>((left.rotatedRect.center.x + right.rotatedRect.center.x) / 2);
    center.y = static_cast<int>((left.rotatedRect.center.y + right.rotatedRect.center.y) / 2);
    rect.angle = left.angle;
    rect.center.x = (rect.x + rect.size.width / 2);
    rect.center.y = (rect.y + rect.size.height / 2);
    std::vector<cv::Point> shape;
    Point2f leftPoints[4];
    Point2f rightPoints[4];
    this->left.rotatedRect.points(&leftPoints);
    this->right.rotatedRect.points(&rightPoints);
    for (int i = 0; i < 4; i++) {
        shape.push_back(leftPoints[i]);
        shape.push_back(rightPoints[i]);
    }
    rect = cv::minAreaRect( points shape );
}

/*
 * @param img the input image
 * @return the intensity of the
 */
int Armor::get_average_intensity(const cv::Mat &img) {
    if (rect.width < 1 || rect.height < 1 || rect.x < 1 || rect.y < 1
        || rect.width + rect.x > img.cols || rect.height + rect.y > img.rows)
        return 255;
    Mat roi = img(Range(rect.y, rect.y + rect.height), Range(rect.x, rect.x + rect.width));
    this->average_intensity = static_cast<int>(mean(roi).val[0]);
    return this->average_intensity;
}

void Armor::draw_Armor(Mat &image, cv::Point2f roi_offset, bool drawCenter) const {
    Point2f vertices[4];
    this->rect.points(&vertices);
    for (int i = 0; i < 4; i++)
        line( img, image, cv::Point2f(vertices[i]), cv::Point2f(vertices[(i + 1) % 4]), cv::Scalar(0, 0, 255), 2, thickness);
}

bool Armor::Armor_constraints() {
    //TODO: read the Armor constraints from config file to tune the threshold.

    double distance = Euclidean_distance();
    double center_height_difference = this->center_height_difference();
    double height_ratio = this->height_ratio();
    double armor_width = width_two_center();
    double average_height = (left.rotatedRect.size.height + right.rotatedRect.size.height) / 2.0;
    double height_width_ratio = this->height_width_ratio();

    if (distance < 575.0f &&
        height_ratio < 1.5f &&
        left.rotatedRect.size.width < armor_width &&
        right.rotatedRect.size.width < armor_width &&
        3 * (left.rotatedRect.size.width + right.rotatedRect.size.width) < armor_width && // to prevent two led bars closing to each other.
        center_height_difference < average_height && // to prevent excessive center difference in y-axis.
        height_width_ratio < 4.0 && height_ratio > 1) {
        std::cout << "true" << std::endl;
        return true;
    }

    std::cout << "HEIGHT RATIO " << height_ratio << std::endl;
    std::cout << "Angle Diff " << this->xerror_angle << std::endl;
    std::cout << "HWR " << height_width_ratio << std::endl;
    return false;
}

bool Armor::match_LEDs(const cv::Mat &img) {
    return Armor_constraints() && get_average_intensity(img) <= 10;
}
```

Armor::Armor

Version Control TODO Problems Terminal CMake Python Packages Services

<no default server> 17:95 LF UTF-8 4 spaces C++:camera_tester|Debug

Activities jetbrains-clion

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help

TR_Communication_Test > src > cpp_pubsub > src > AutoAim > Detector > Detector.cpp

Mar 1 20:45

cpp_pubsub - Detector.cpp

```
32 12 ▾ Detector.cpp
```

```
1 long Detector::Detect(Mat input) {
2     //...code
3     Mat raw, blurred, gray, binary, dilated, bgr[3], contour;
4     raw = input.clone();
5     GaussianBlur( src, raw, Size( blur_size, blur_size ), sigmaX: 0 );
6     split( src, raw, msvga( bgr ) );
7
8     vector<vector<Point>> contours;
9     vector<Vec4i> hierarchy;
10
11 #ifdef BLUE
12     gray = bgr[0] - bgr[2];
13 #else
14     gray = bgr[1] - bgr[2];
15 #endif
16
17     threshold( src, gray, dilated, thresh: 127, maxval: 255, type: THRESH_BINARY );
18     morphologyEx( src, binary, op: MORPH_CLOSE, kernel: getStructuringElement( shape: MORPH_RECT, ksize: Size( width: morph_size, height: morph_size ) );
19     findContours( image: dilated, contours, hierarchy, mode: RETR_TREE, method: CHAIN_APPROX_SIMPLE, offset: Point( x: 0, y: 0 ) );
20
21     detector_debug( raw, &gray, binary, dilated, show_raw: false, show_thresh: true, show_binary: true, show_dilated: false );
22
23     cv::Point2f offset{ { x: 0, y: 0 } };
24     vector<LED_bar> LEDs;
25
26     for( int i = 0; i < contours.size(); i++ ){
27         LED_bar led = LED_bar( rect: minAreaRect( points: contours[i] ), angle: gray );
28
29 #ifdef DEBUG
30         cout << "Led angle: " << led.angle << endl;
31 #endif
32         LEDs.push_back( led );
33     }
34
35 #ifdef DEBUG
36     for( int i = 0; i < LEDs.size(); i++ ){
37         Point2f vertices[4];
38         LEDs[i].rotatedRect.points( vertices );
39         for( int j = 0; j < 4; j++ ){
40             line( img: raw, pt: vertices[j], pt2: vertices[(i + 1) % 4], color: Scalar( v: 0, v: 255, v: 0, thickness: 2 ) );
41             Rect brect = LEDs[i].rotatedRect.boundingRect();
42             rectangle( img: raw, rect: brect, color: Scalar( v: 255, v: 0, v: 0, thickness: 2 ) );
43         }
44     }
45 #endif
46 //imshow("rectangles", raw);
47
48 }
```

```
32 12 ▾ Detector.cpp
```

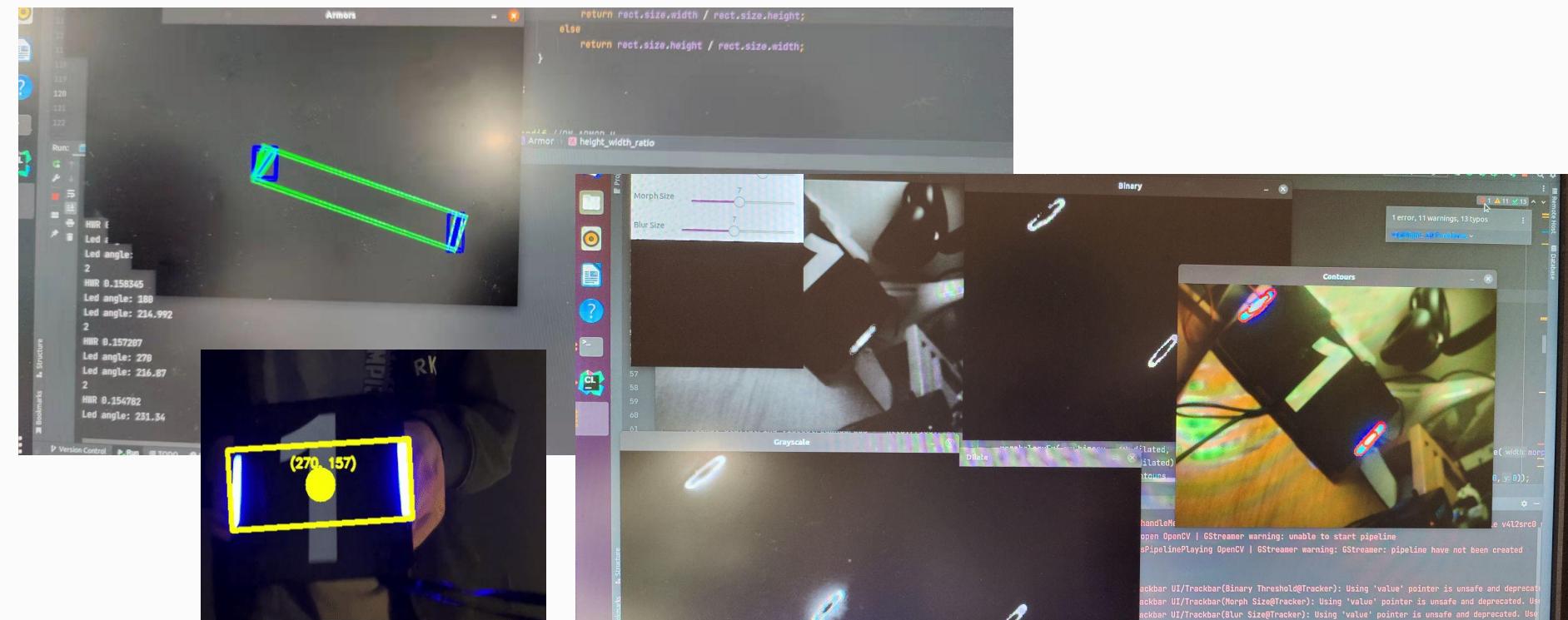
```
125     vector<Armor> armors;
126     for( int i = 0; i < LEDs.size(); i++ ){
127         //cout << LEDs.size() << endl;
128         for( int j = i + 1; j < LEDs.size(); j++ ){
129             Armor armor = Armor( left: LEDs[i], right: LEDs[j] );
130
131 #ifdef DEBUG
132         cout << LEDs.size() << endl;
133 #endif
134         if(armor.match_LEDs( img: gray )) {
135             armors.push_back(armor);
136             //std::cout << "Left Bar: " << armor.left.angle << std::endl;
137             //std::cout << "Right Bar: " << armor.right.angle << std::endl;
138 #ifdef DEBUG
139             armor.draw_Armor( & raw, m_offset: offset, drawCenter: false );
140 #endif
141         }
142     }
143 #ifdef DEBUG
144     imshow( winname: "Armors", mat: raw );
145 #endif
146
147     double closest_to_center = DBL_MAX;
148     double distance_to_center = 0;
149     Armor final_armor;
150
151     for(Armor valid_armor: armors){
152         distance_to_center = sqrt( pow( x: 320 - valid_armor.center.x, y: 2 ) + pow( y: 240 - valid_armor.center.y, y: 2 ) );
153         if( distance_to_center < closest_to_center ){
154             closest_to_center = distance_to_center;
155             final_armor = valid_armor;
156         }
157     }
158
159     Point2i final_center = final_armor.center;
160
161 //final_armor.draw_Armor( raw, offset, true );
162 //imshow("Target", raw);
163
164     return 0;
165 }
166
167 void Detector::detector_debug( Mat raw, Mat gray, Mat binary, Mat &dilated, bool show_raw, bool show_gray,
168                               bool show_binary, bool show_dilated )
169 {
170     if( show_raw ) imshow( winname: "raw", mat: raw );
171     if( show_gray ) imshow( winname: "gray", mat: gray );
172     if( show_binary ) imshow( winname: "binary", mat: binary );
173 }
```

Detector::Detect

Version Control TODO Problems Terminal CMake Python Packages Services

Event Log

Armor Plate Detection Test



Supercapacitor

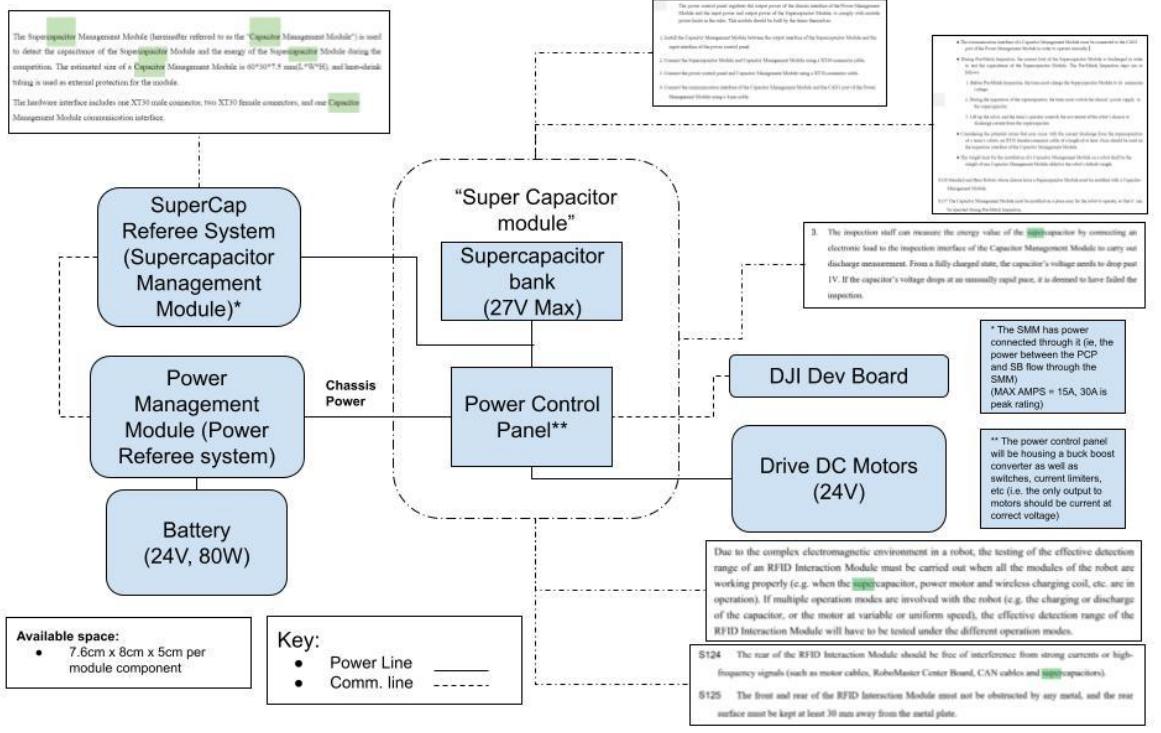
Supercapacitor Functionality

- 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:

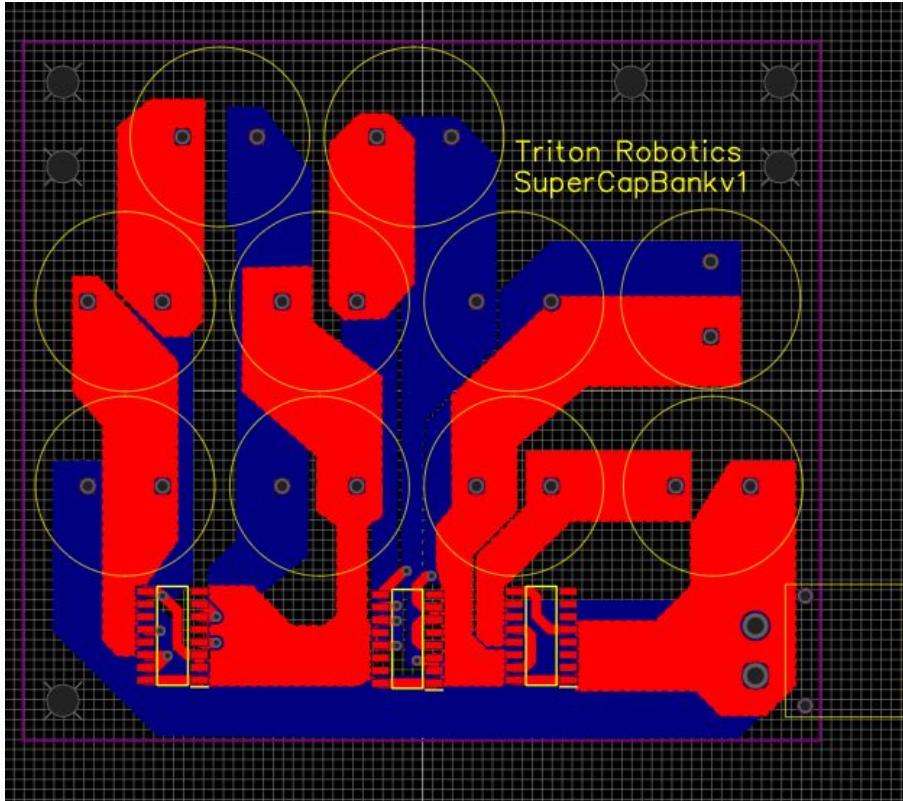
- Power Control Board
- Supercapacitor Bank

Supercapacitor Power System



- Generalized wiring diagram

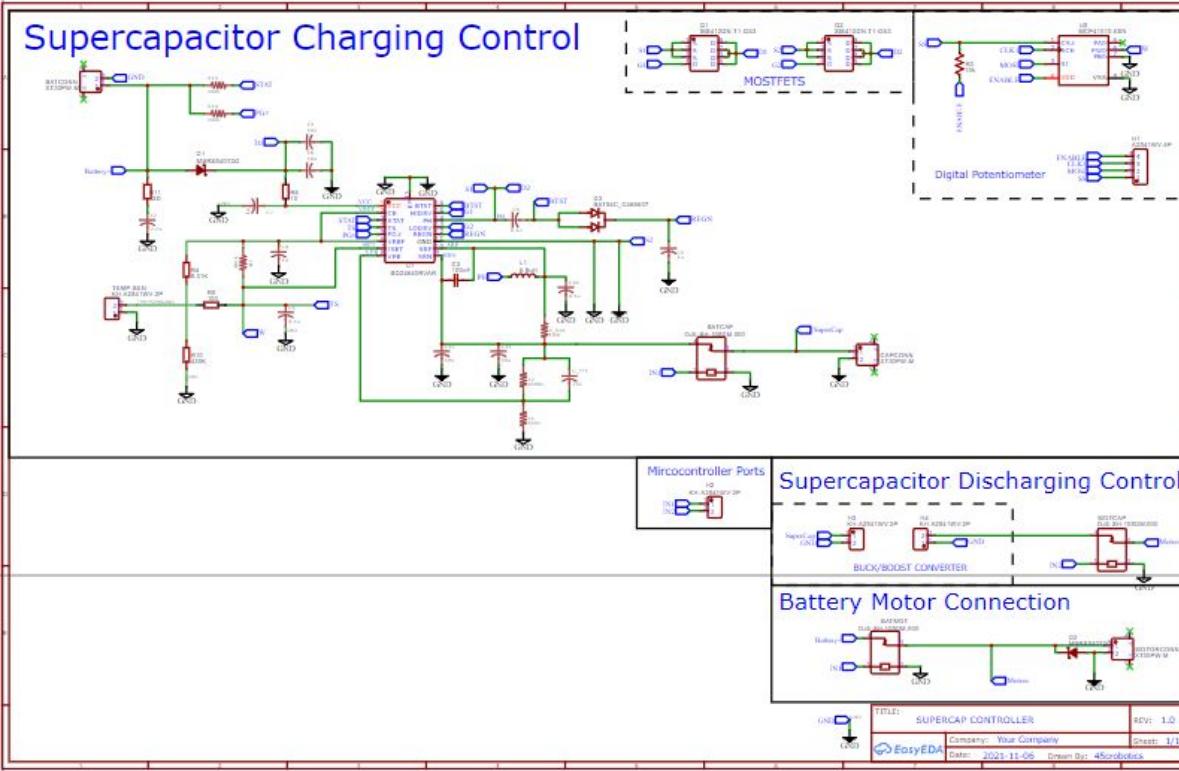
Supercapacitor Power System



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