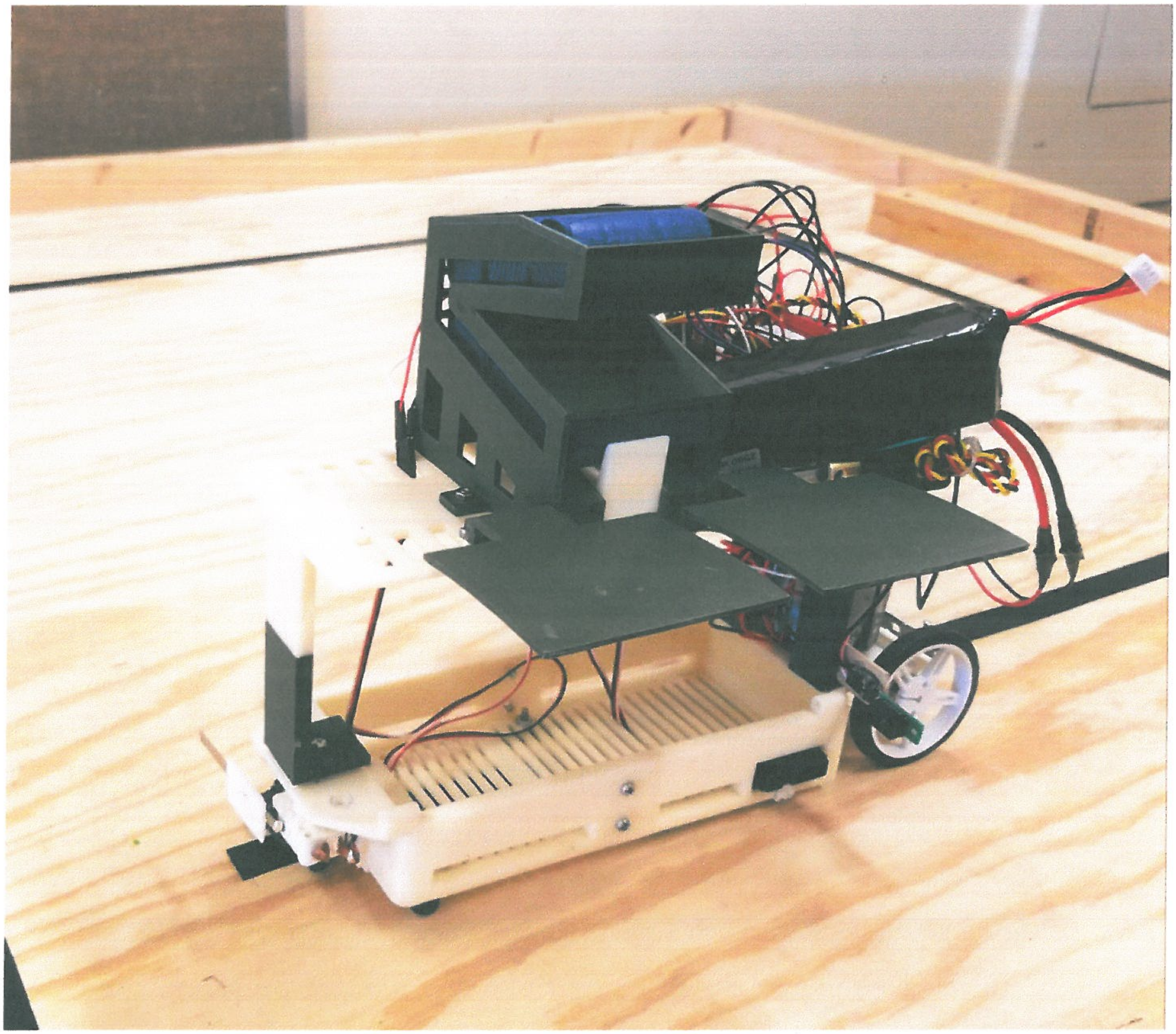
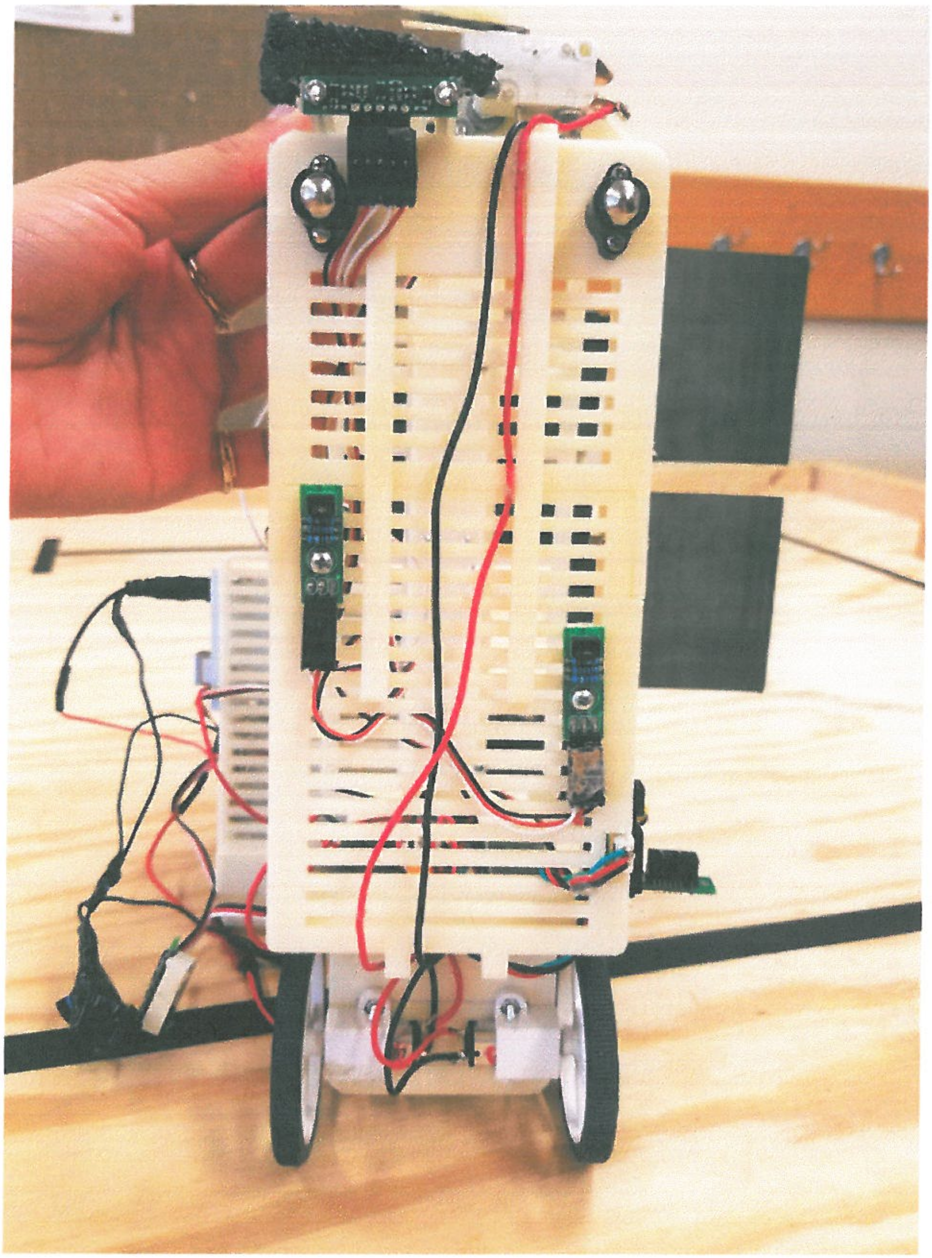
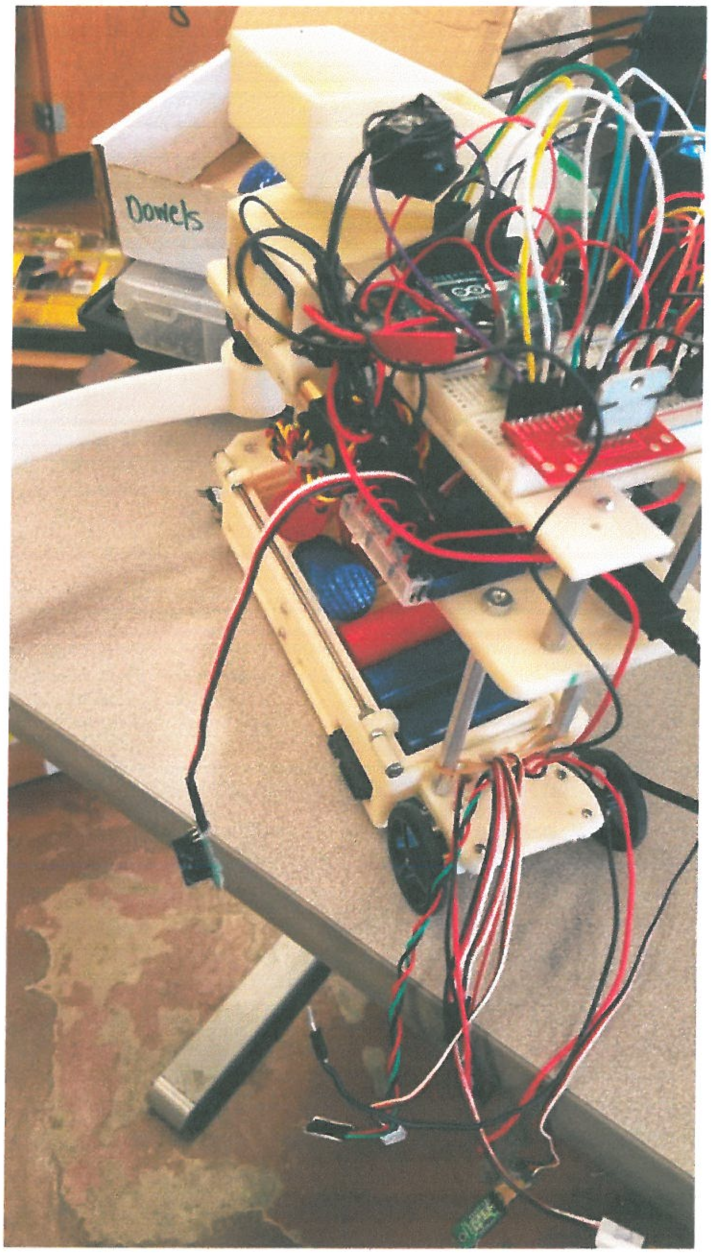
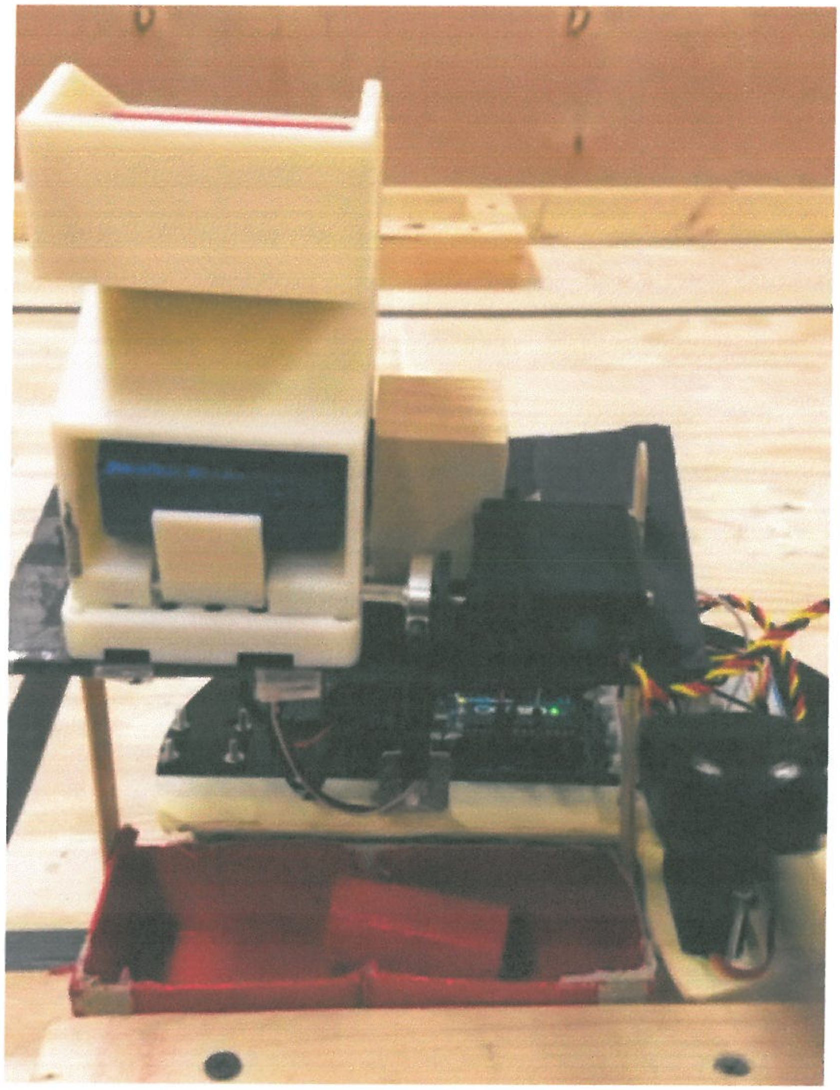
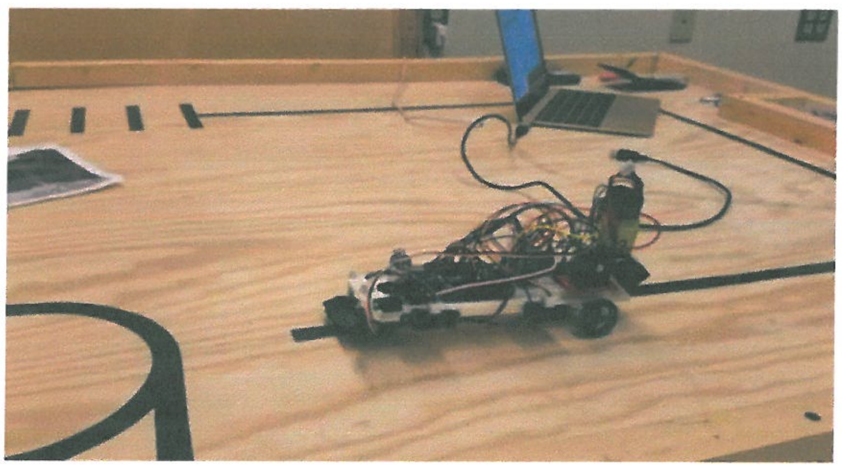
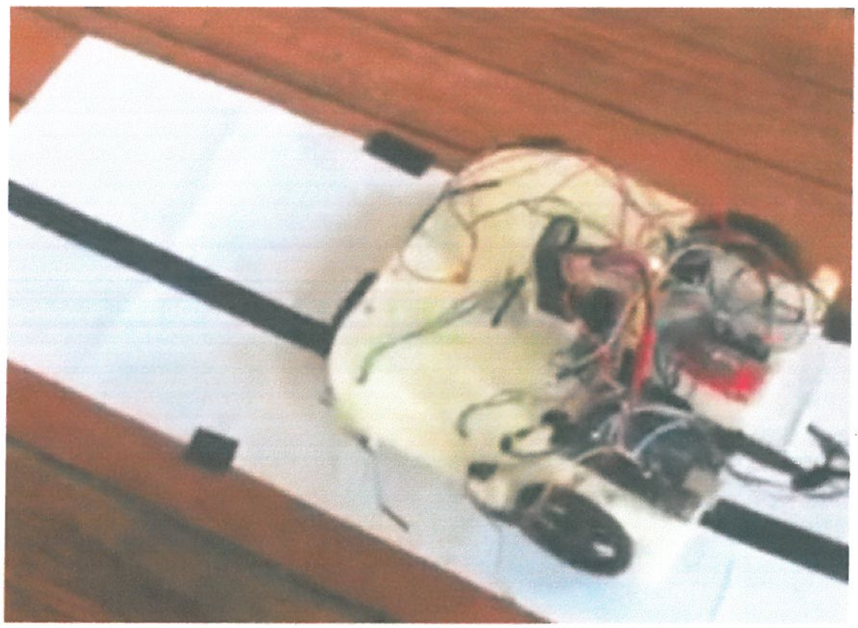


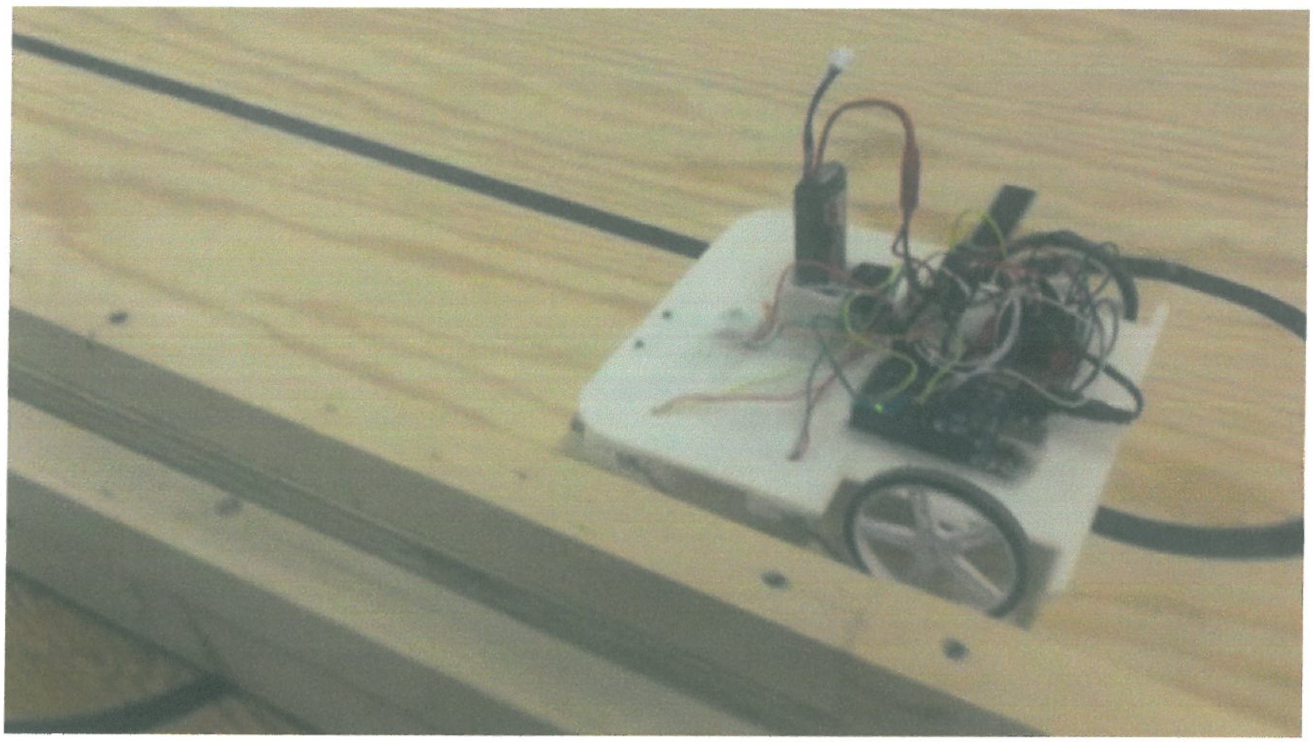


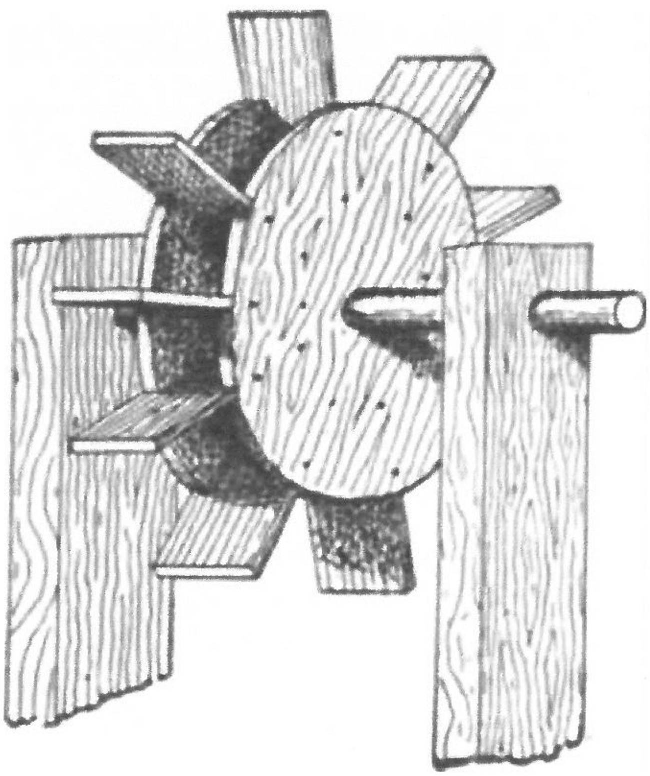

Dowels

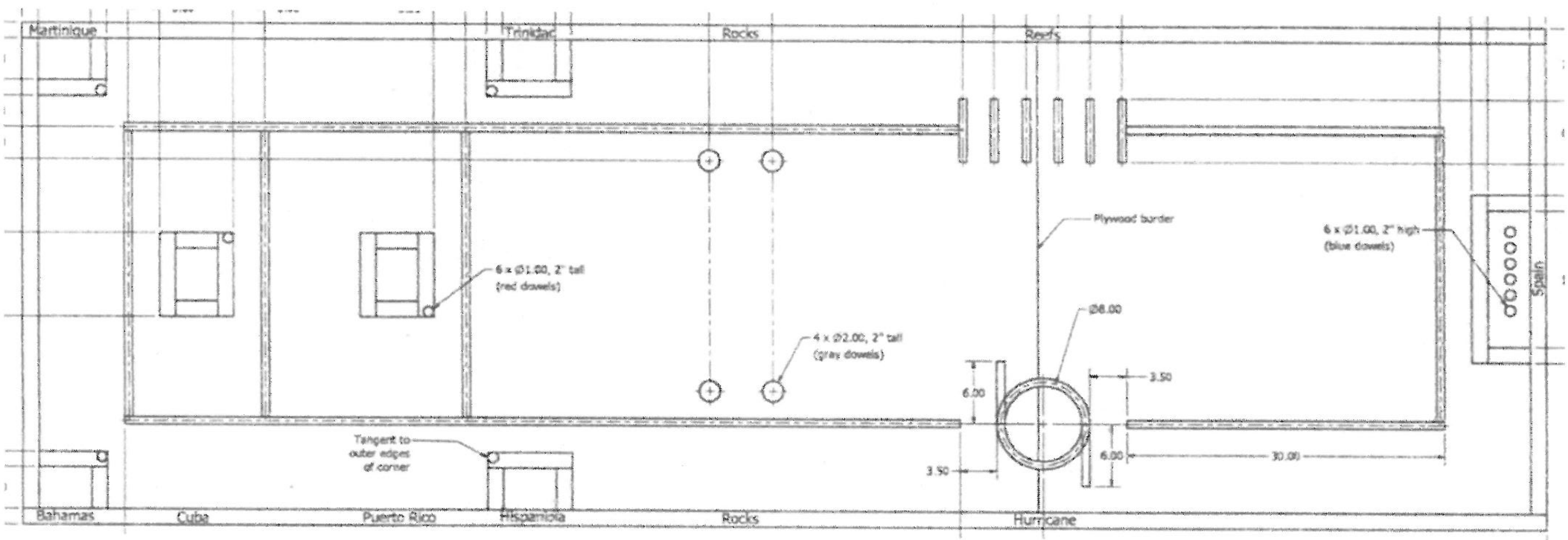




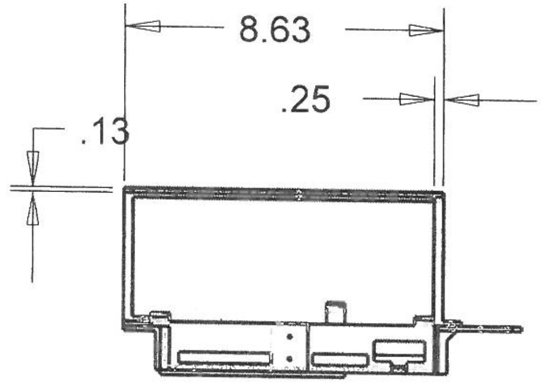


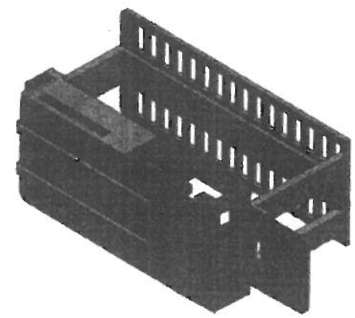


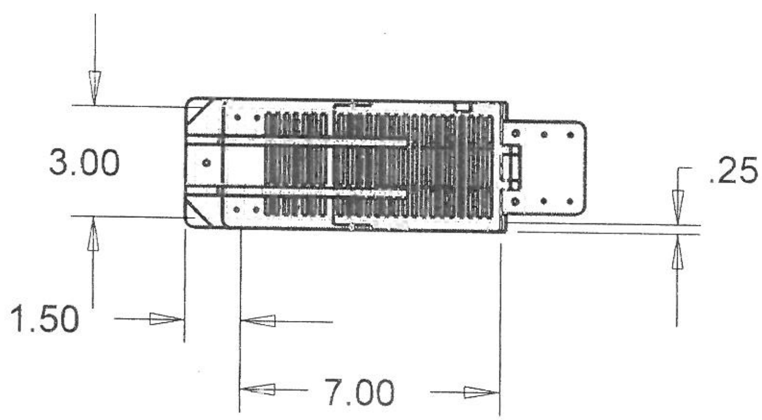


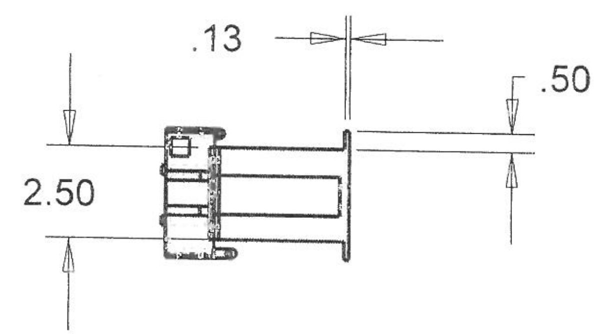


Martinique
Trinidad
Rocks
Reefs
Plywood border
6 x Ø1.00, 2" high
(blue dowels)
Spain
6 x Ø1.00, 2" tall
(red dowels)
Ø8.00
4 x Ø2.00, 2" tall
(gray dowels)
3.50
6.00
6.00
30.00
3.50
Tangent to
outer edges
of corner
Bahamas
Cuba
Puerto Rico
Hispaniola
Rocks
Hurricane

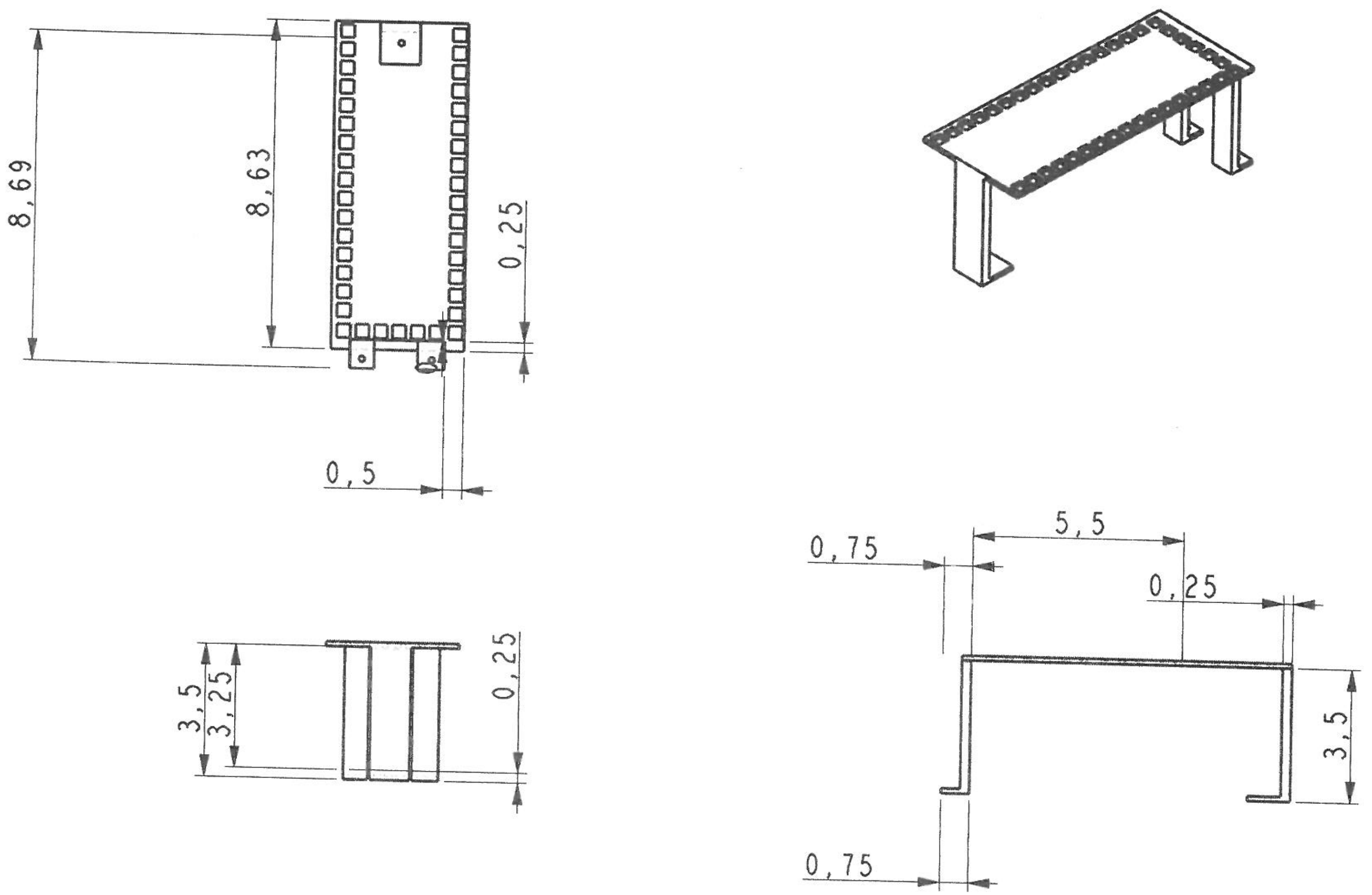




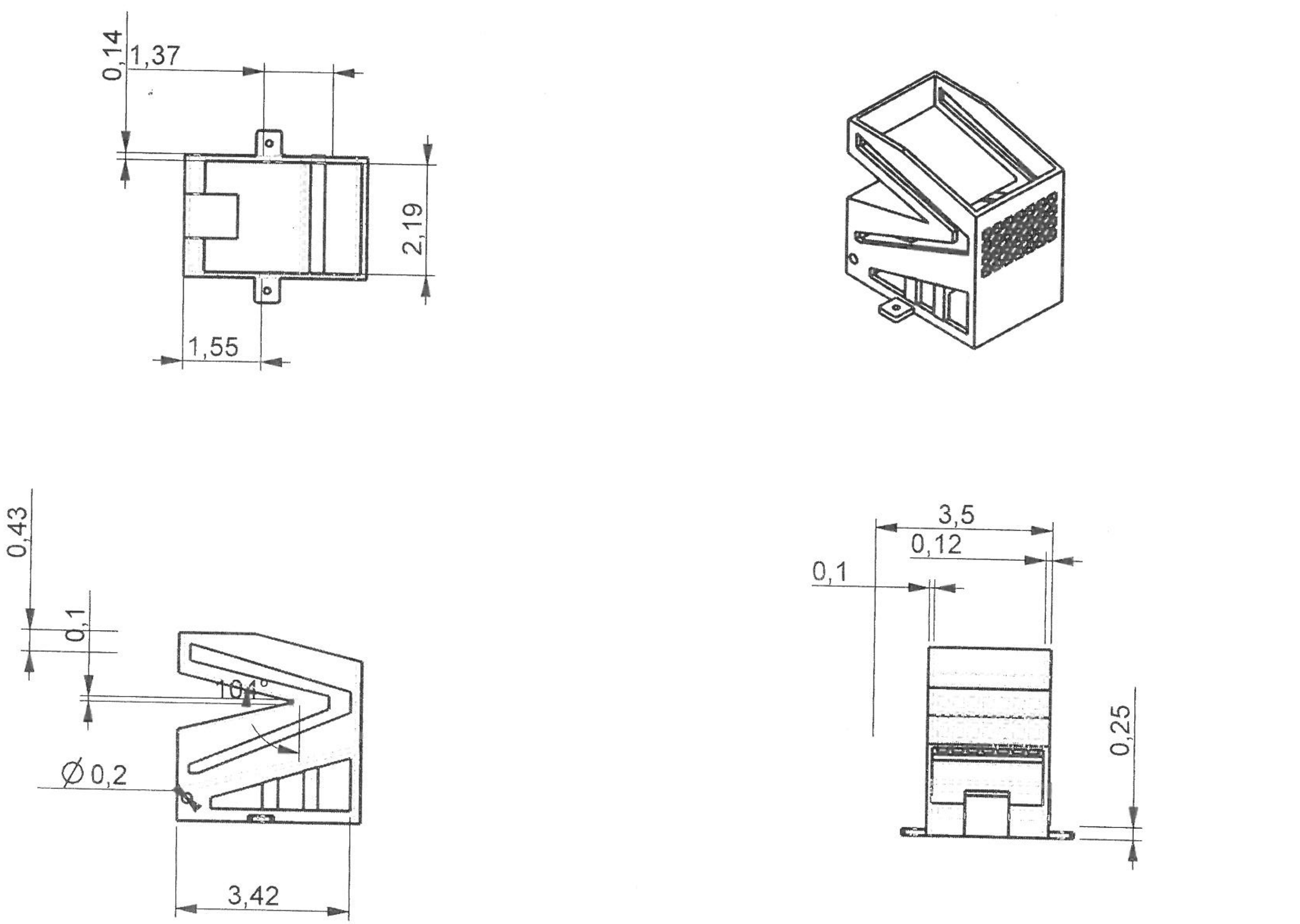




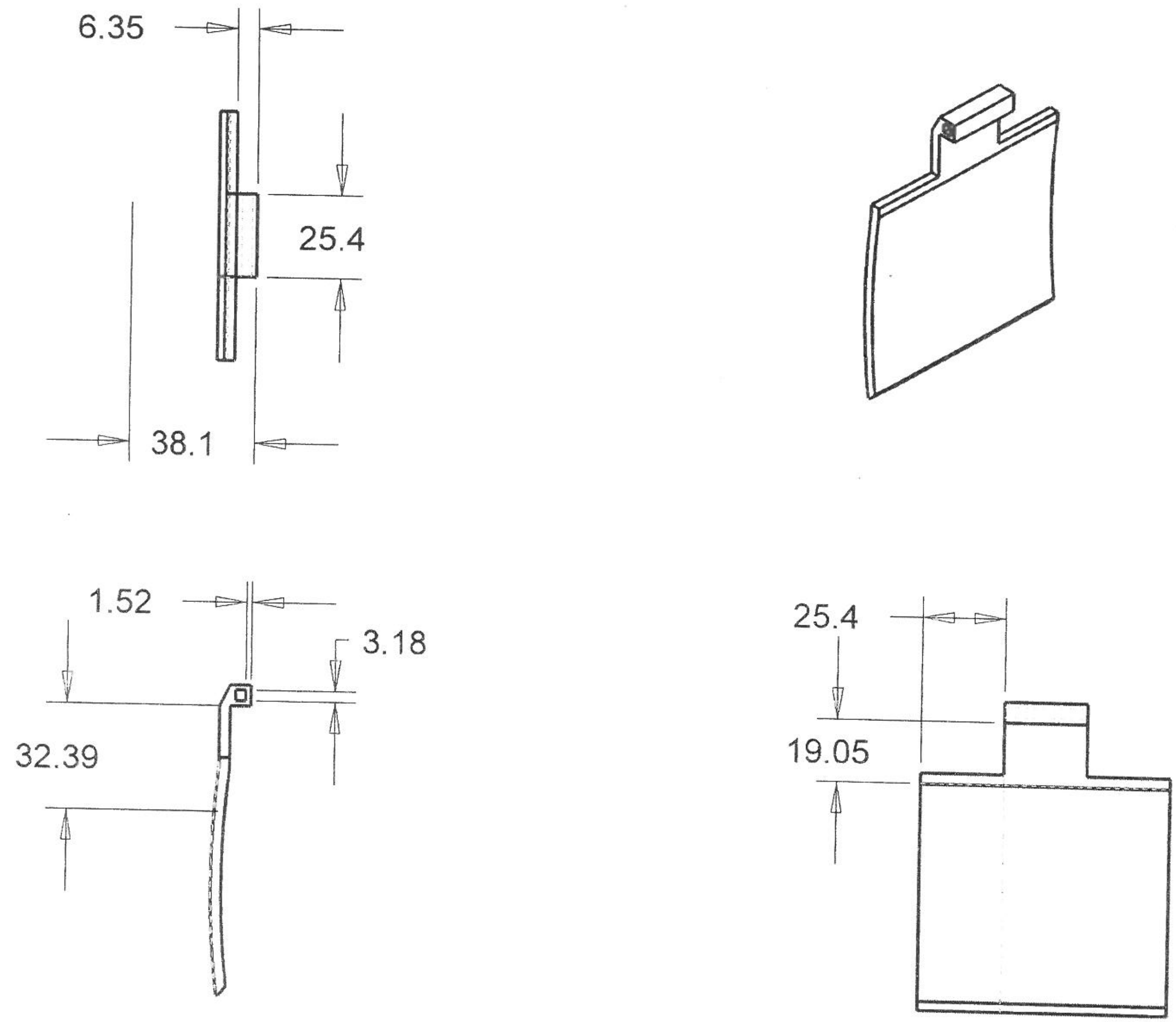
BMCC Robotic team (2016-2017)
Lead Designer : Edward Valdez
Assistant Designer: Krongchai Praponpoj



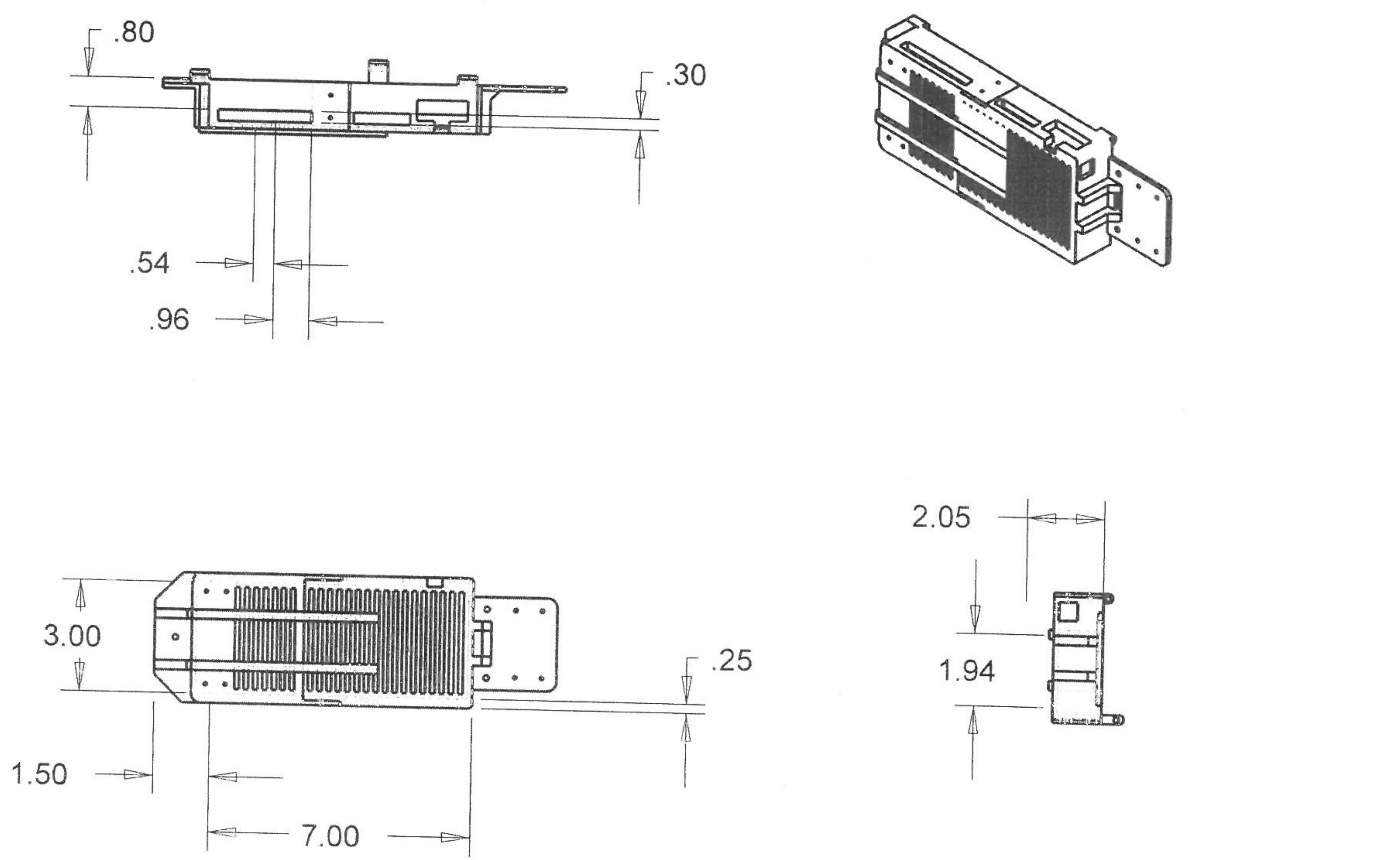
BMCC Robotics Team (2016-2017)
Lead Designer: Edward Valdez
Assistant Designer :Krongchai Praponpoj



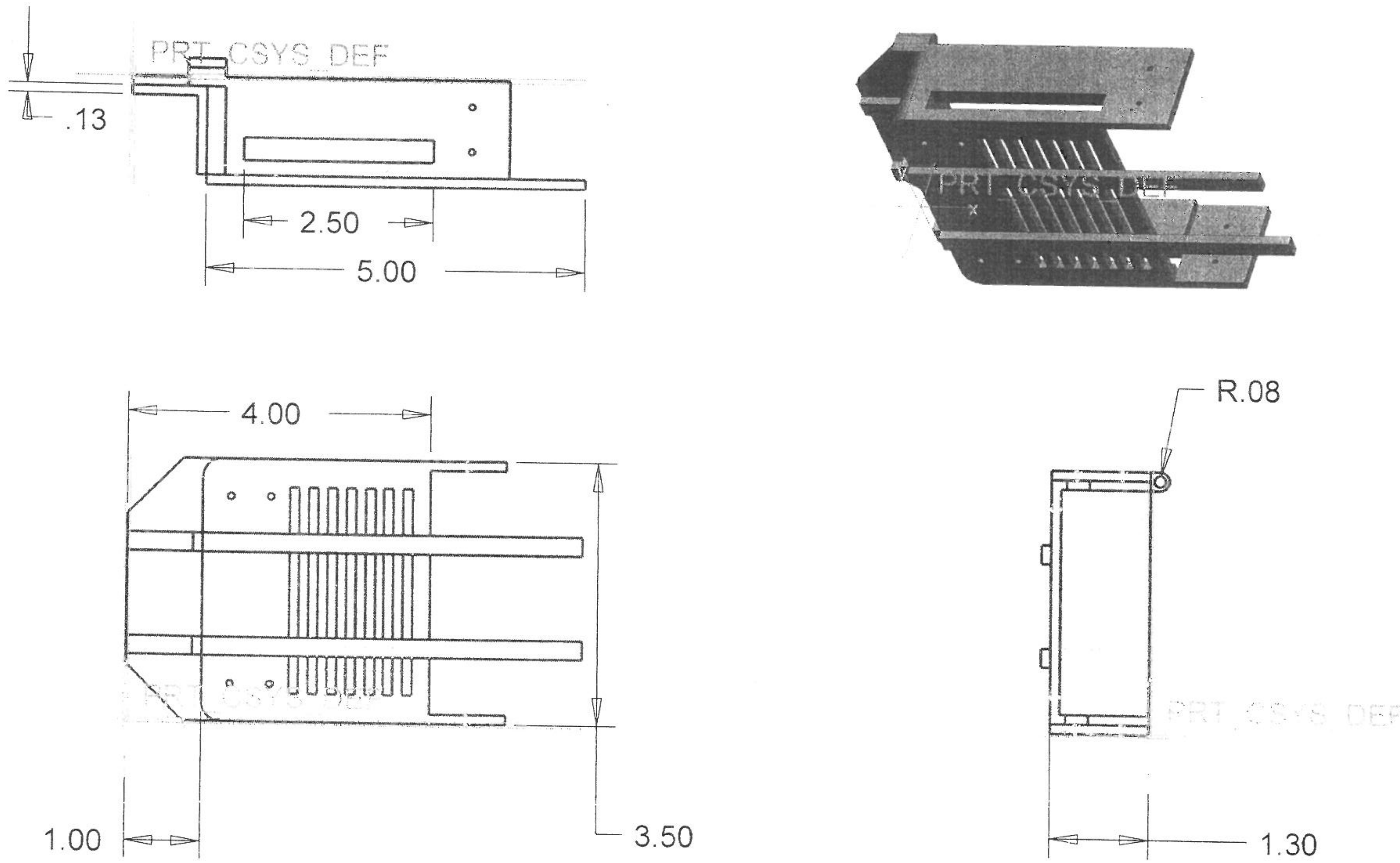
BMCC Robotics Team (2016-2017)
Lead Designer: Edward Valdez
Assistant Designer :Krongchai Praponpoj



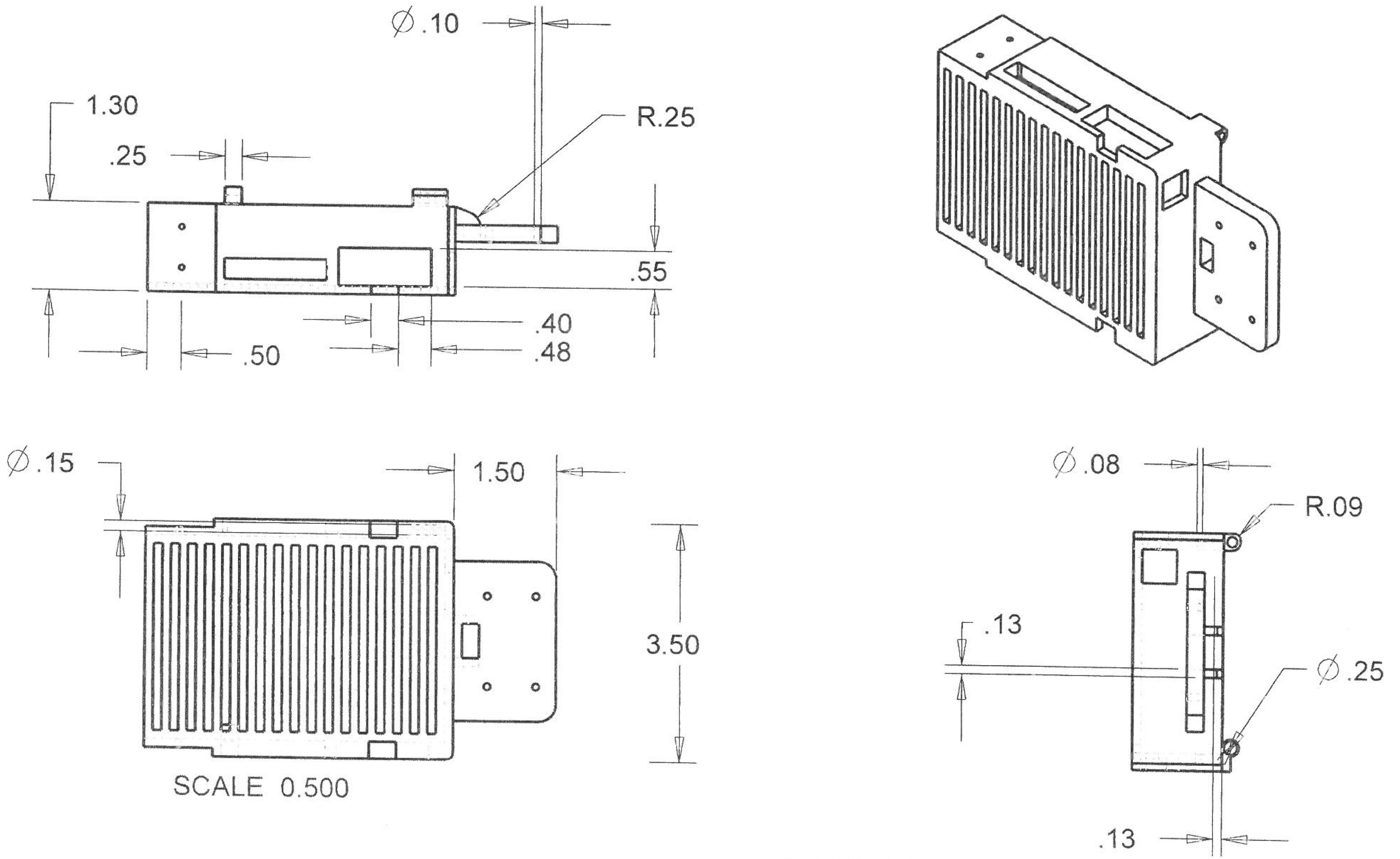
BMCC Robotic team (2016-2017)
Lead Designer : Edward Valdez
Assistant Designer: Krongchai Praponpoj



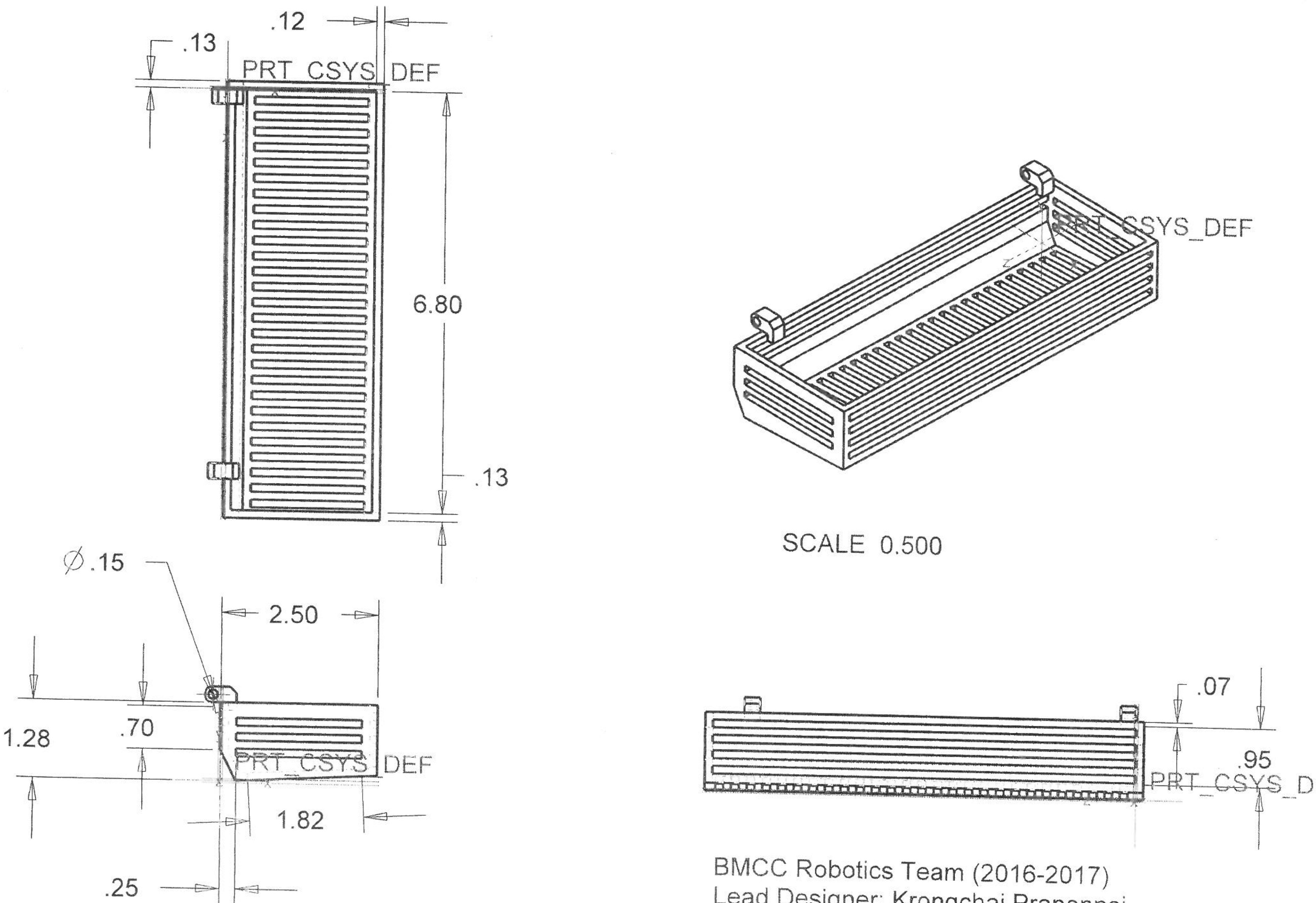
BMCC Robotic team (2016-2017)
Lead Designer : Edward Valdez
Assistant Designer: Krongchai Praponpoj



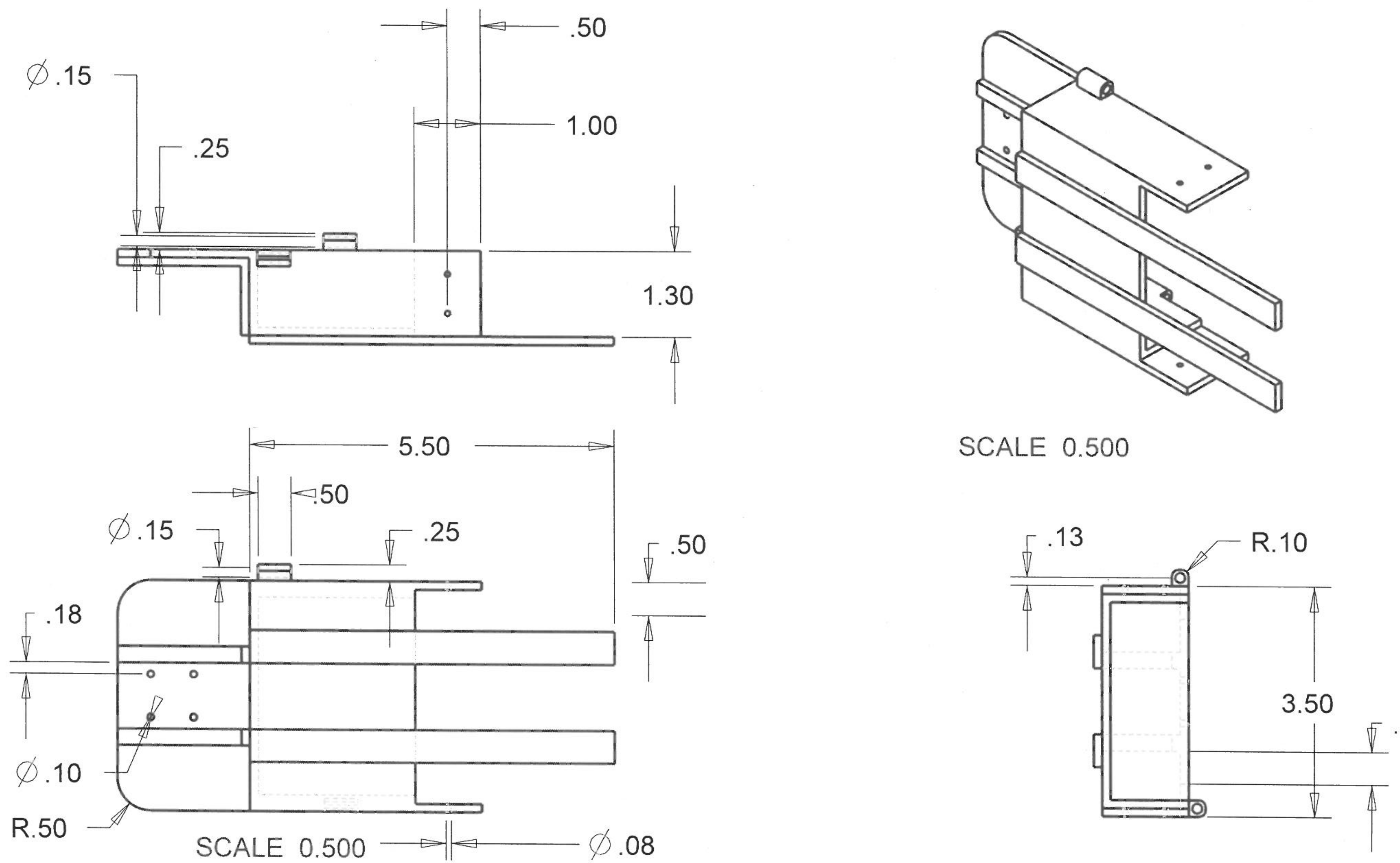
BMCC Robotic team (2016-2017)
Lead Designer : Krongchai Praponpoj
Assistant Designer : Edward Valdez



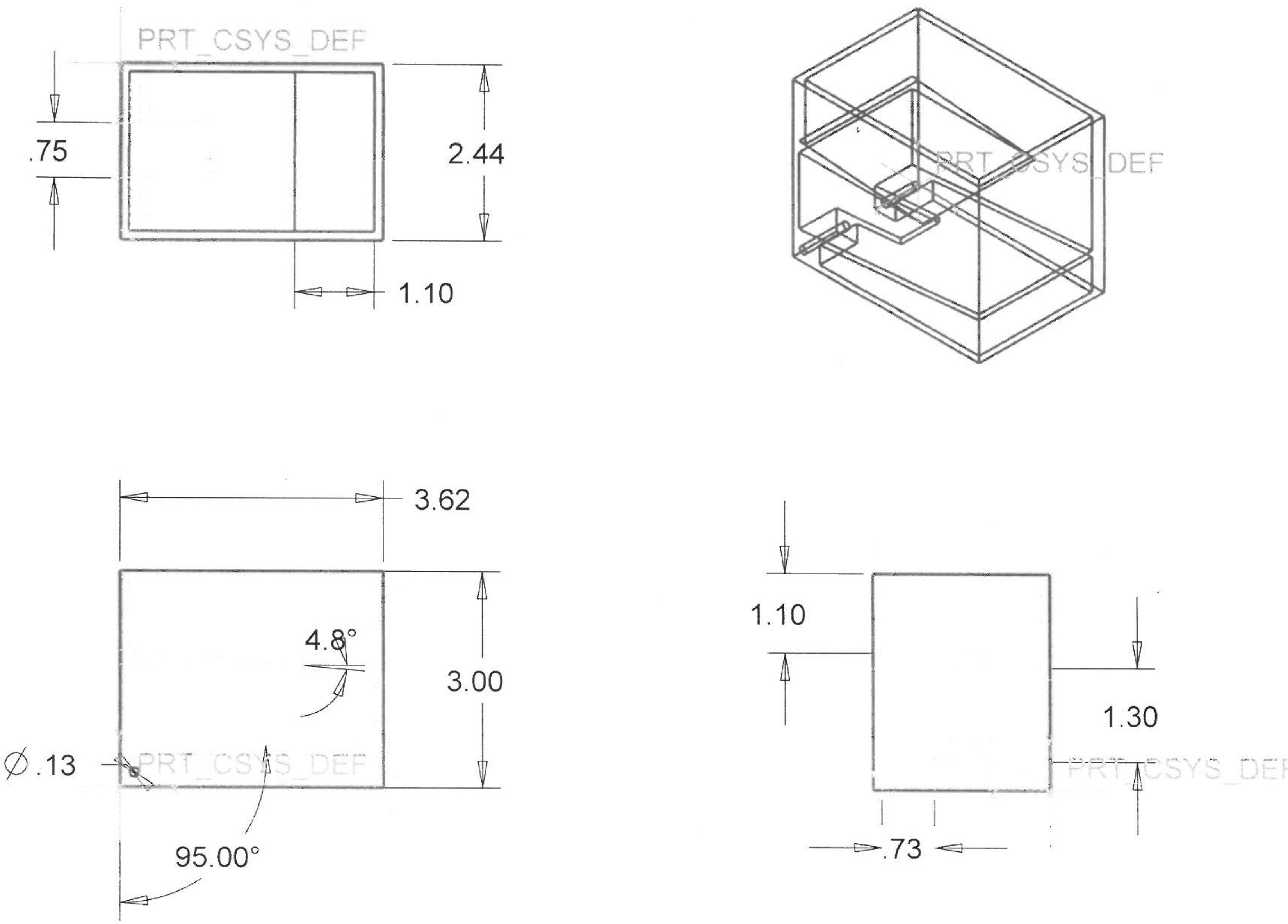
BMCC Robotics Team (2016-2017)
Lead Designer: Krongchai Praponpoj
Assistant Designer: Edward Valdez



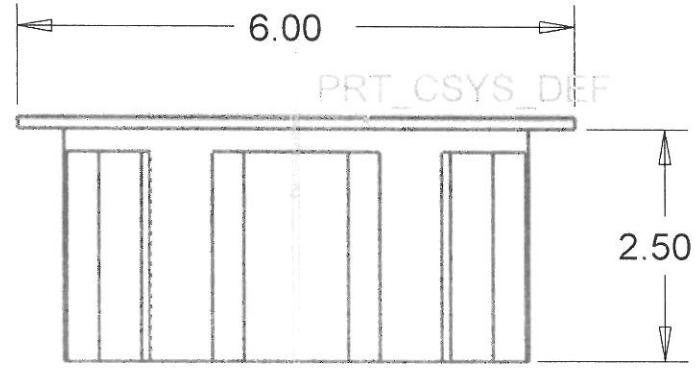
BMCC Robotics Team (2016-2017)
Lead Designer: Krongchai Praponpoj
Assistant Designer: Edward Valdez

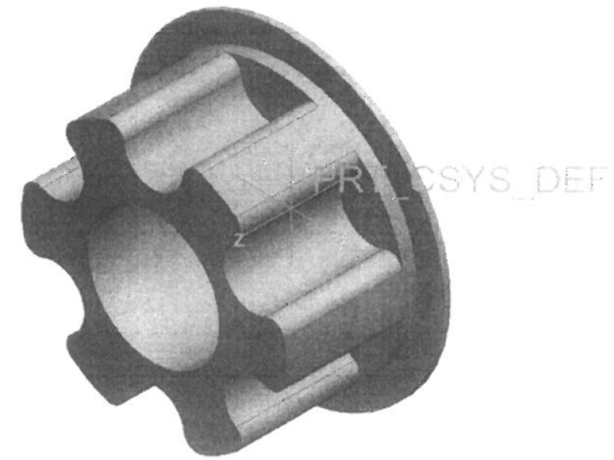


BMCC Robotics Team (2016-2017)
Lead Designer: Krongchai Praponpoj
Assistant Designer: Edward Valdez

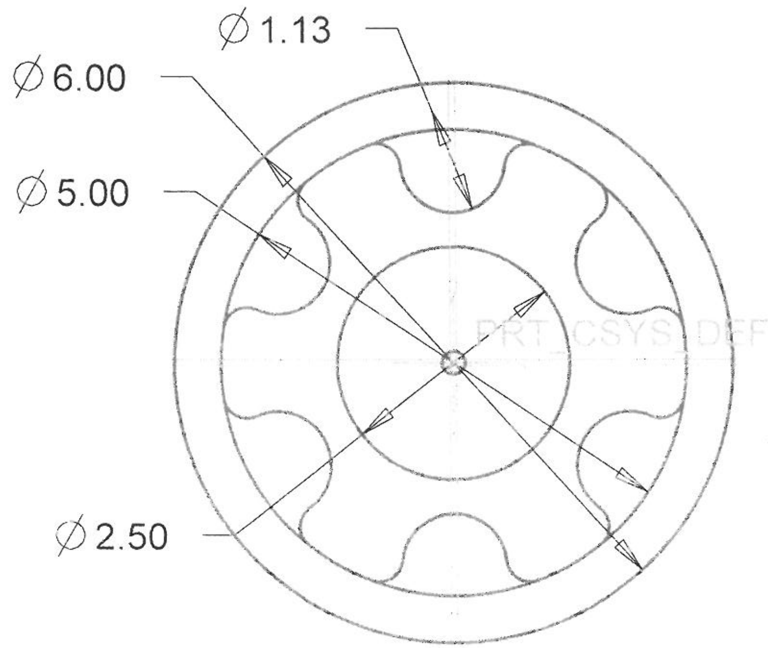


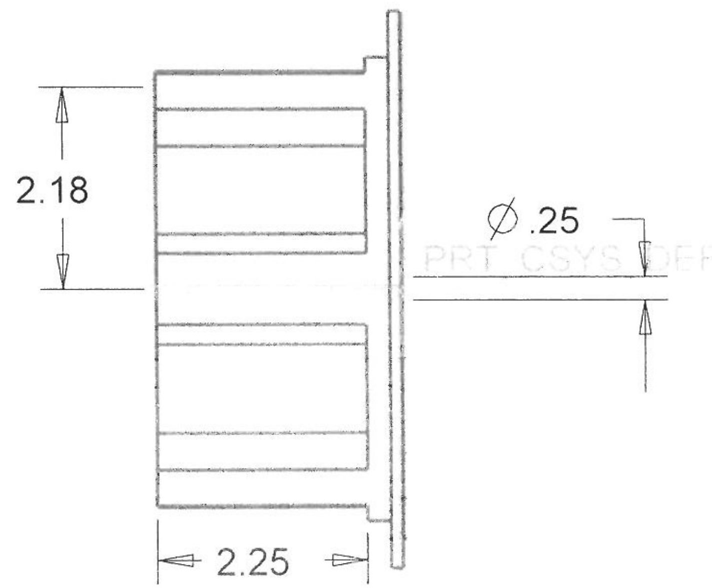
BMCC Robotic Team (2016 - 2017)
Lead Designer : Krongchai Praponpoj
Assitanct Designer : Edward Valdez



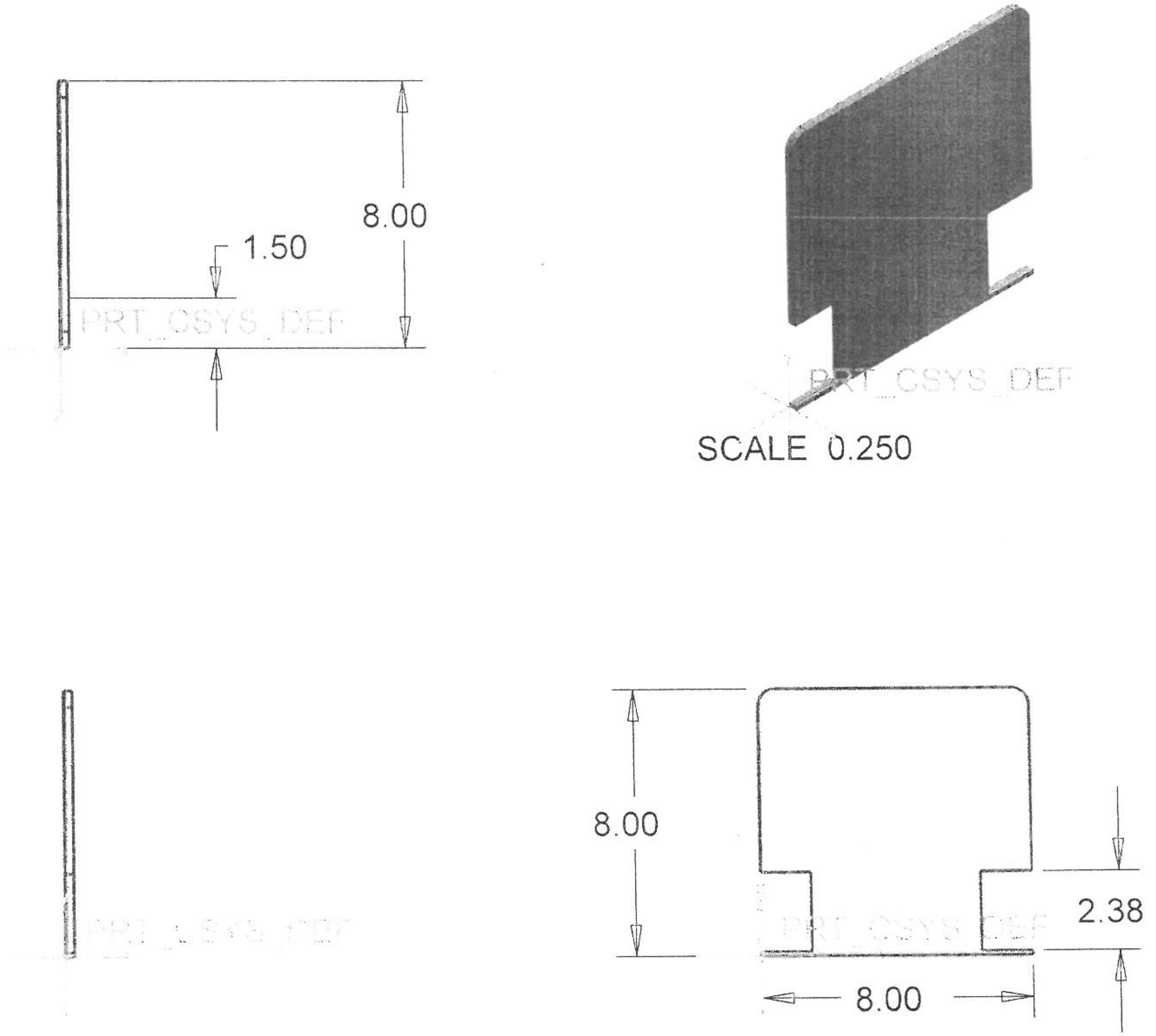


SCALE 0.400



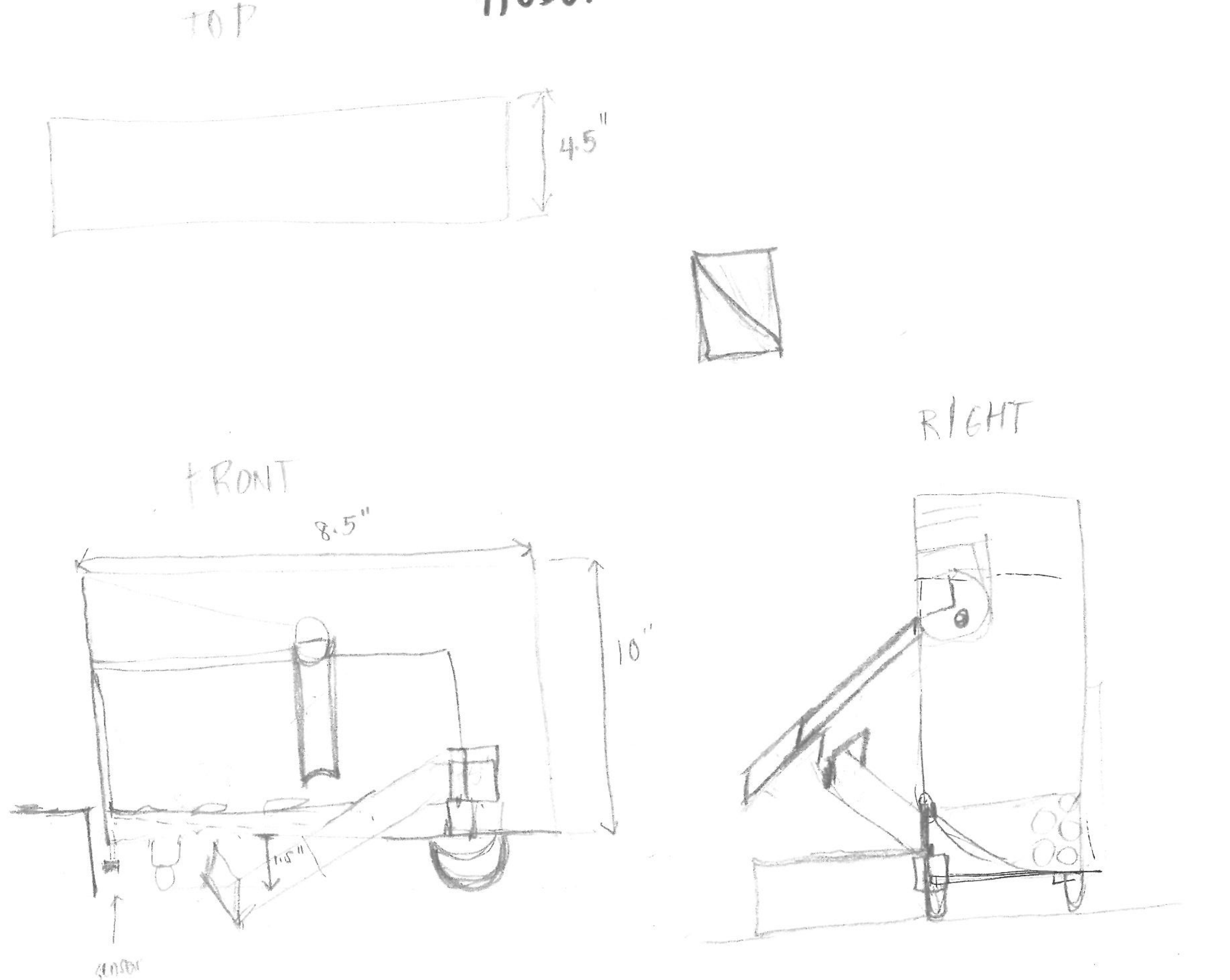


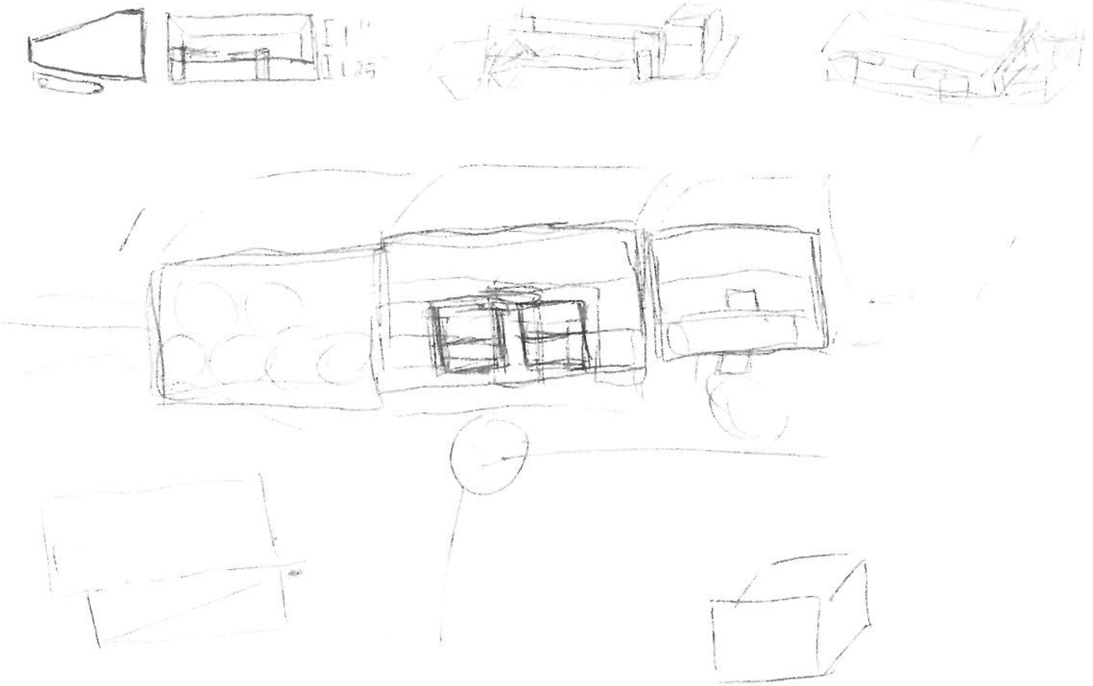
BMCC Robotic Team (2016-2017)
Lead Designer : Edward Valdez
Assistant Designer :
Carlos Clemente
Krongchai Praponpoj



BMCC Robotic Team (2016-2017)
Lead Designer : Edward Valdez
Assistant Designer : Carlos Clemente

Robot # 4 Double vecker bus

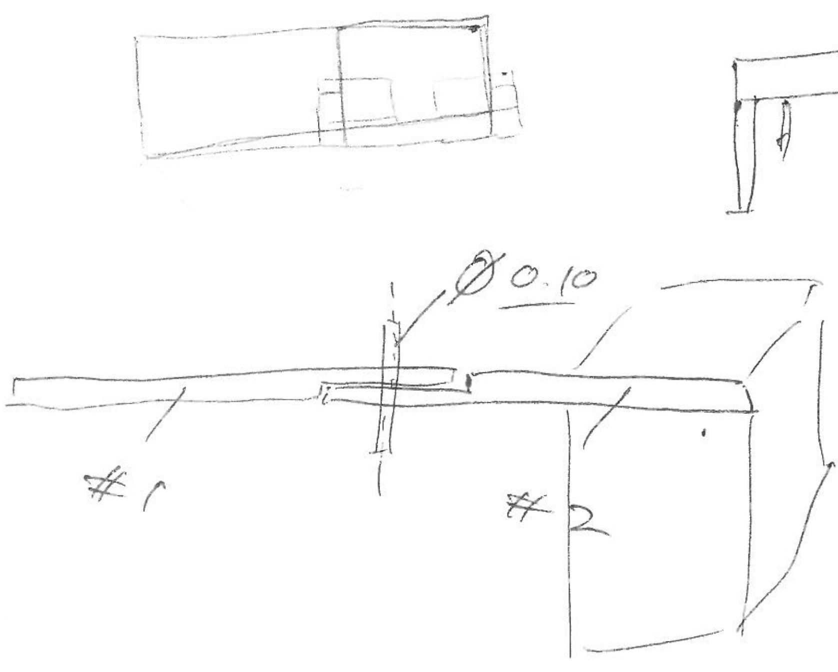


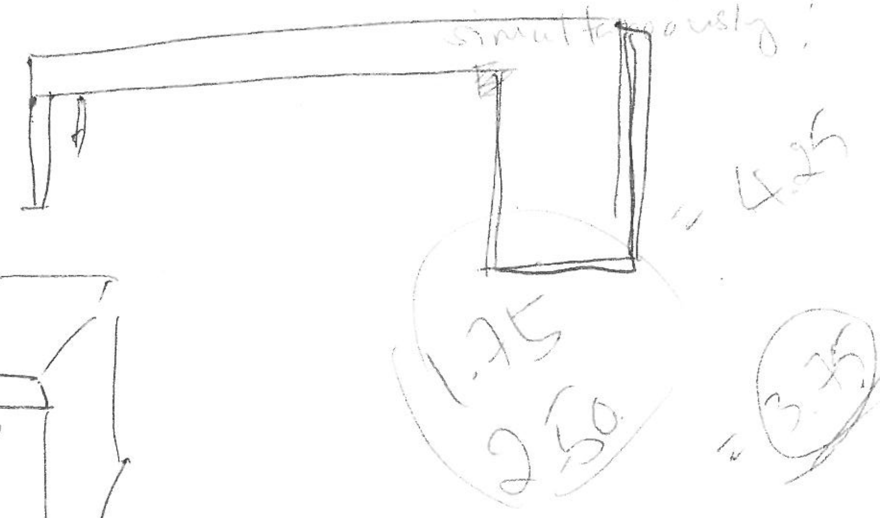


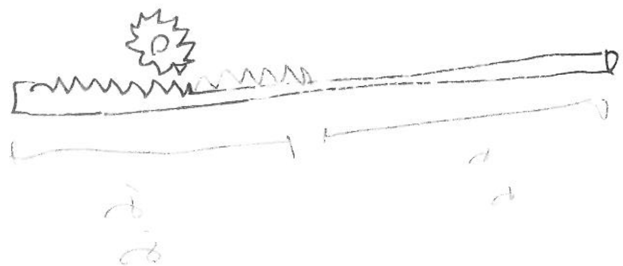
- Side load and dispense

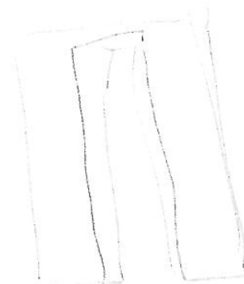
- Side load & Front dispense

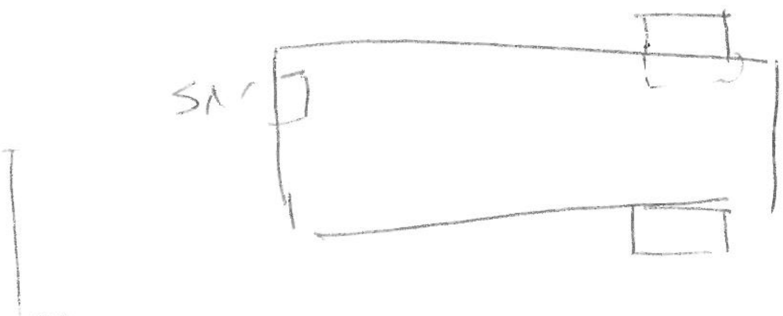
same thing that can simultaneously?

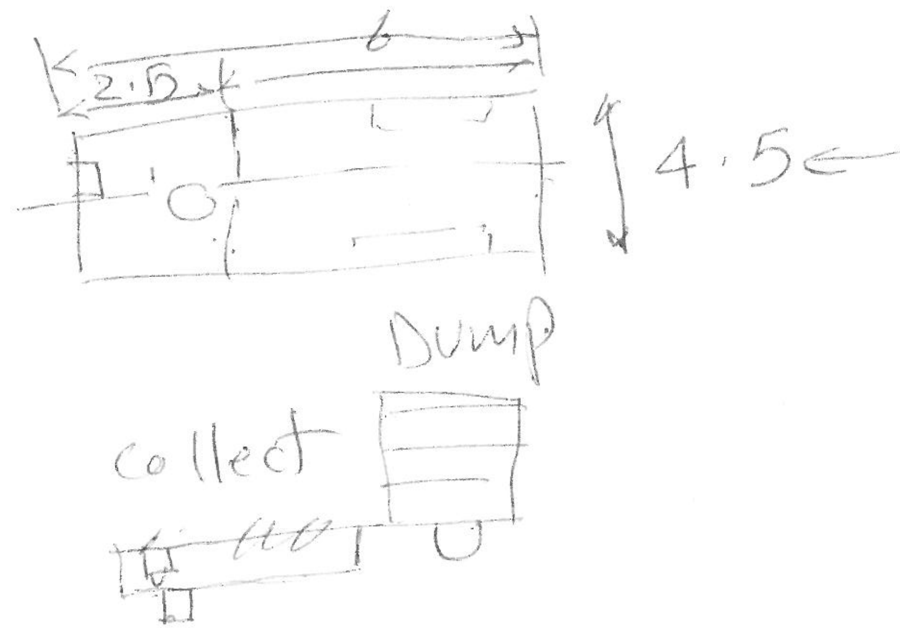




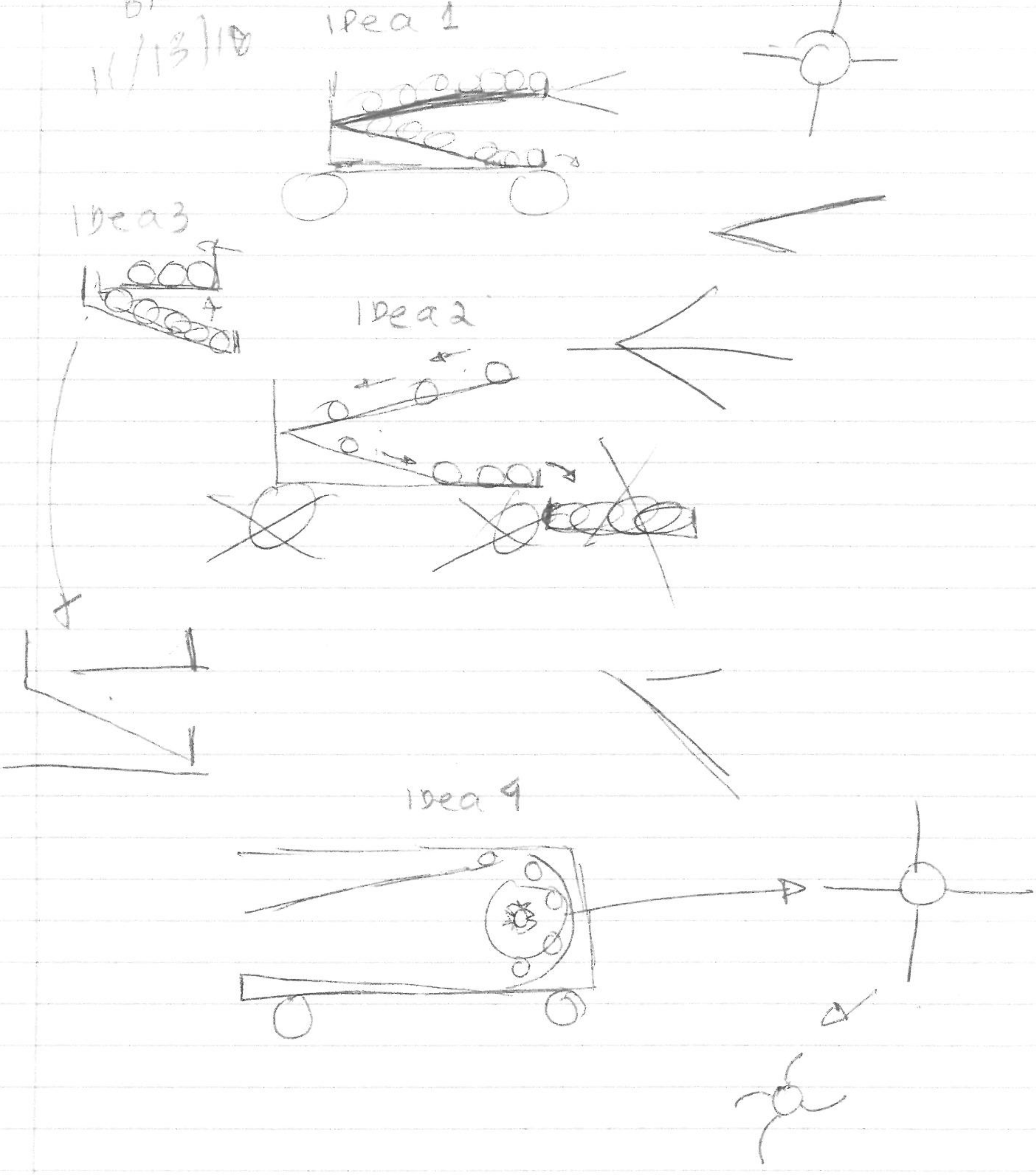


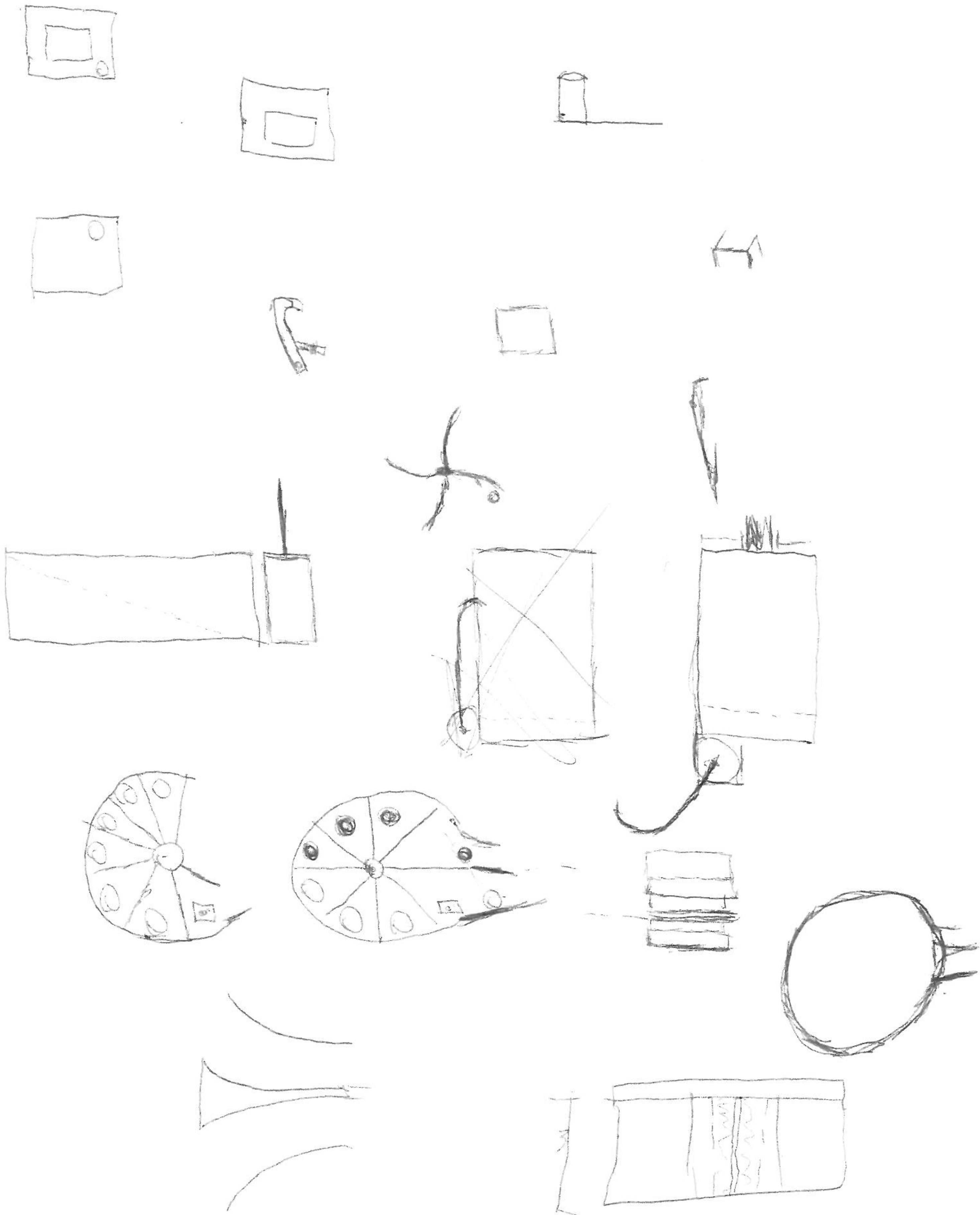




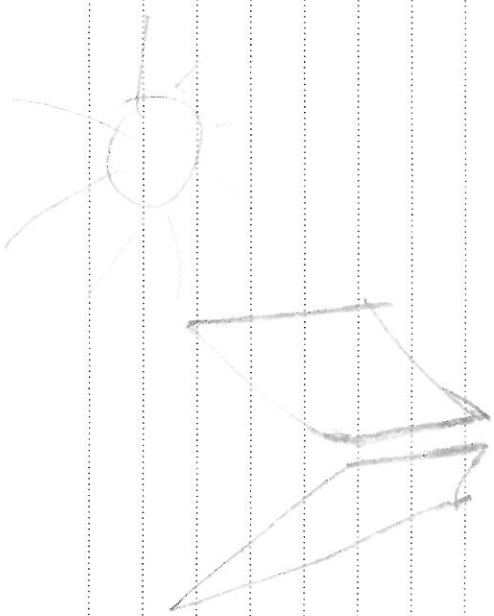

6
2.15
4.5
Dump
collect

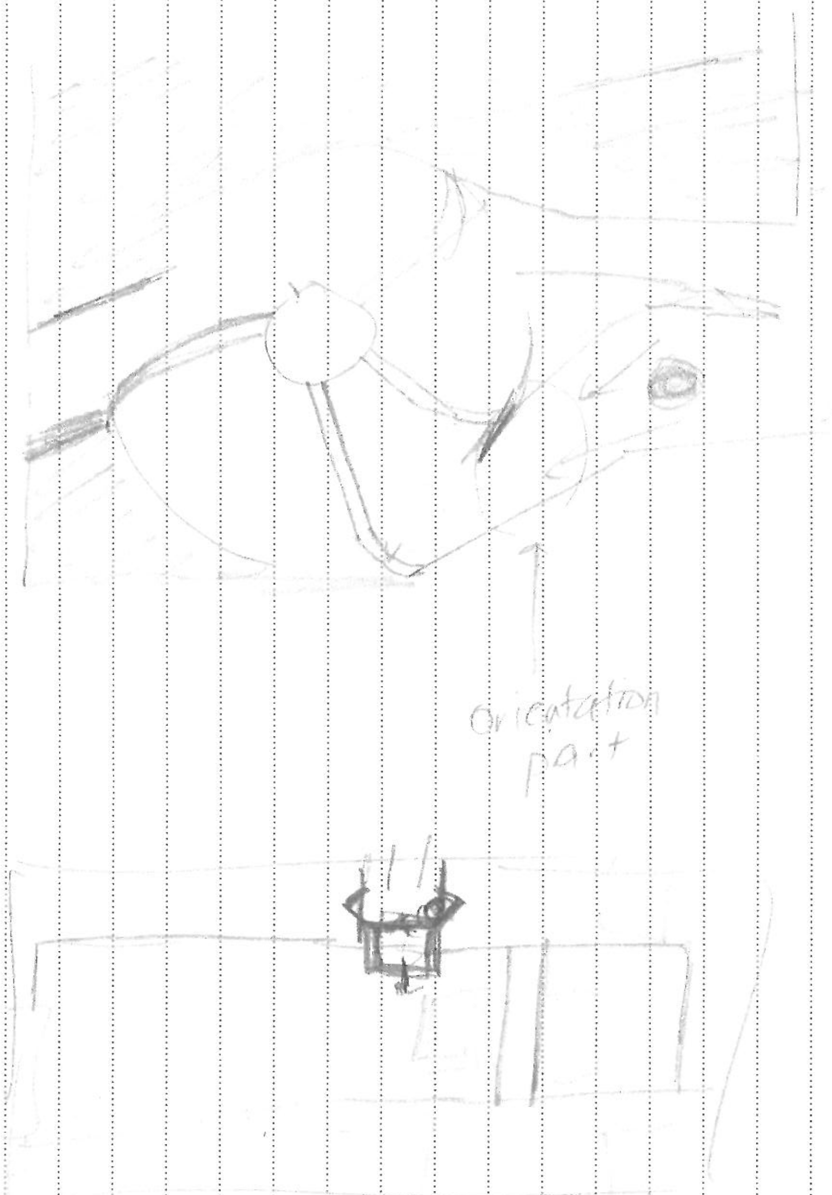
RODRIGO
BRAGA
11/13/18



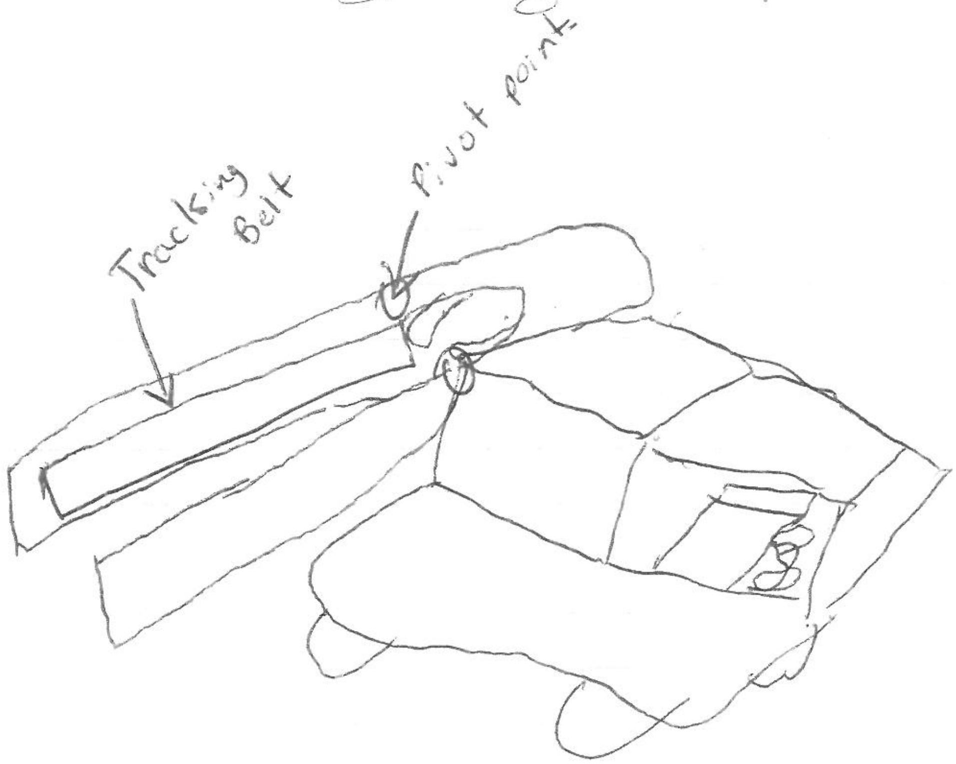


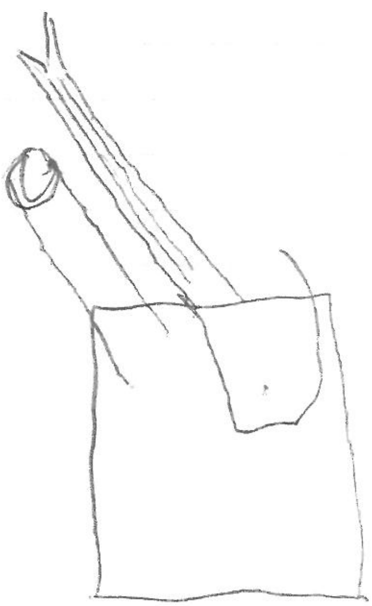
Efren roman@live.com

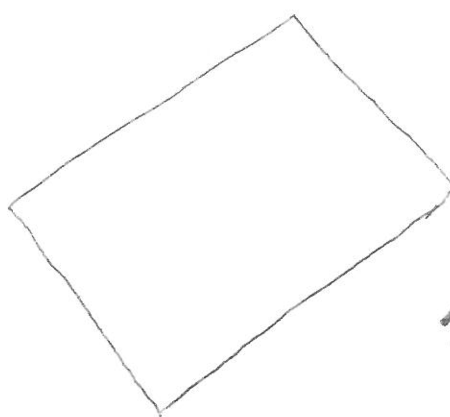


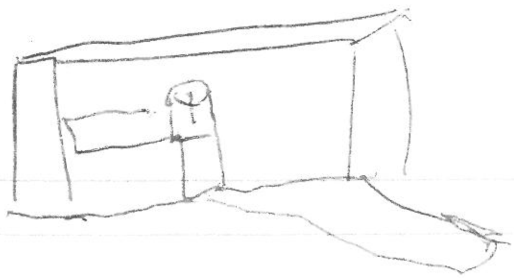

Orientation
part

The Barbeque Master

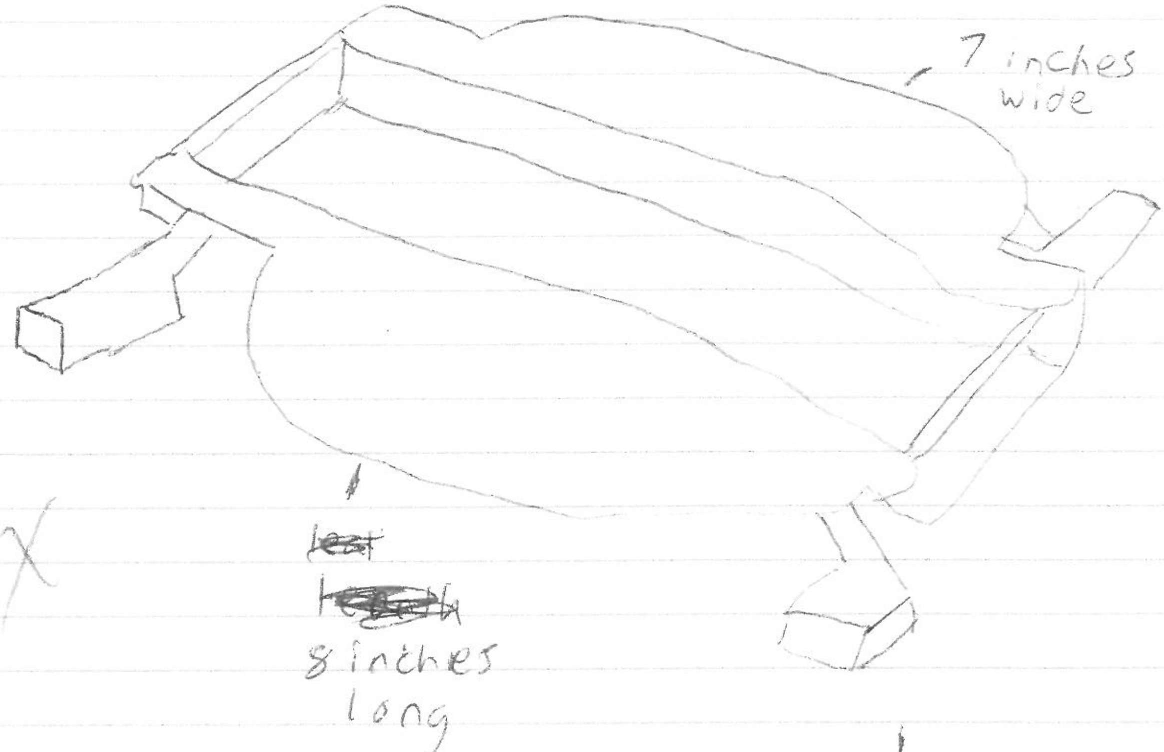






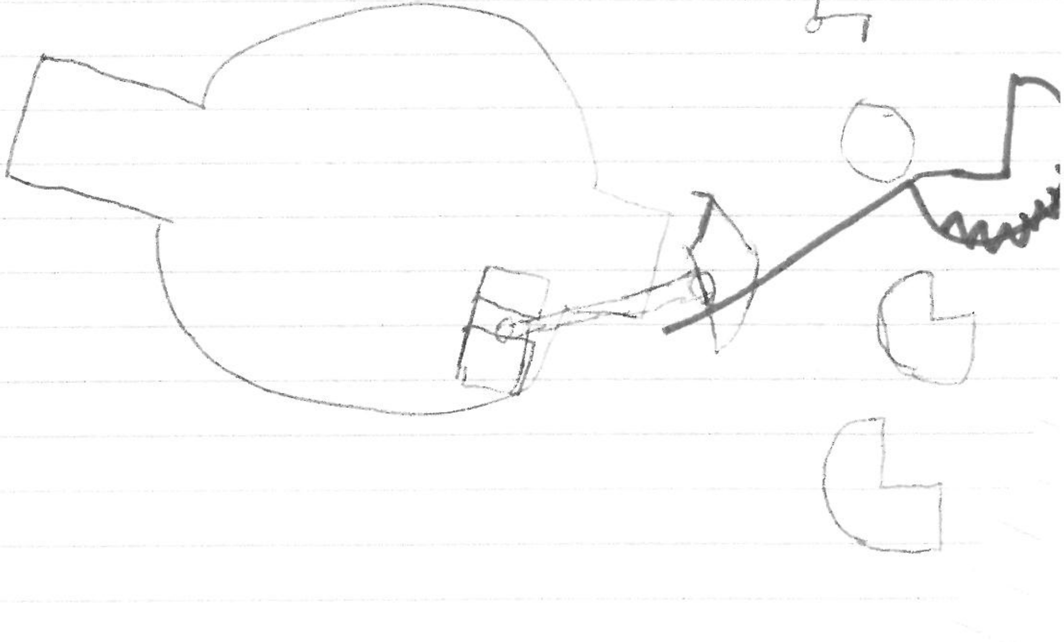


Edward Valdez
11/13/16



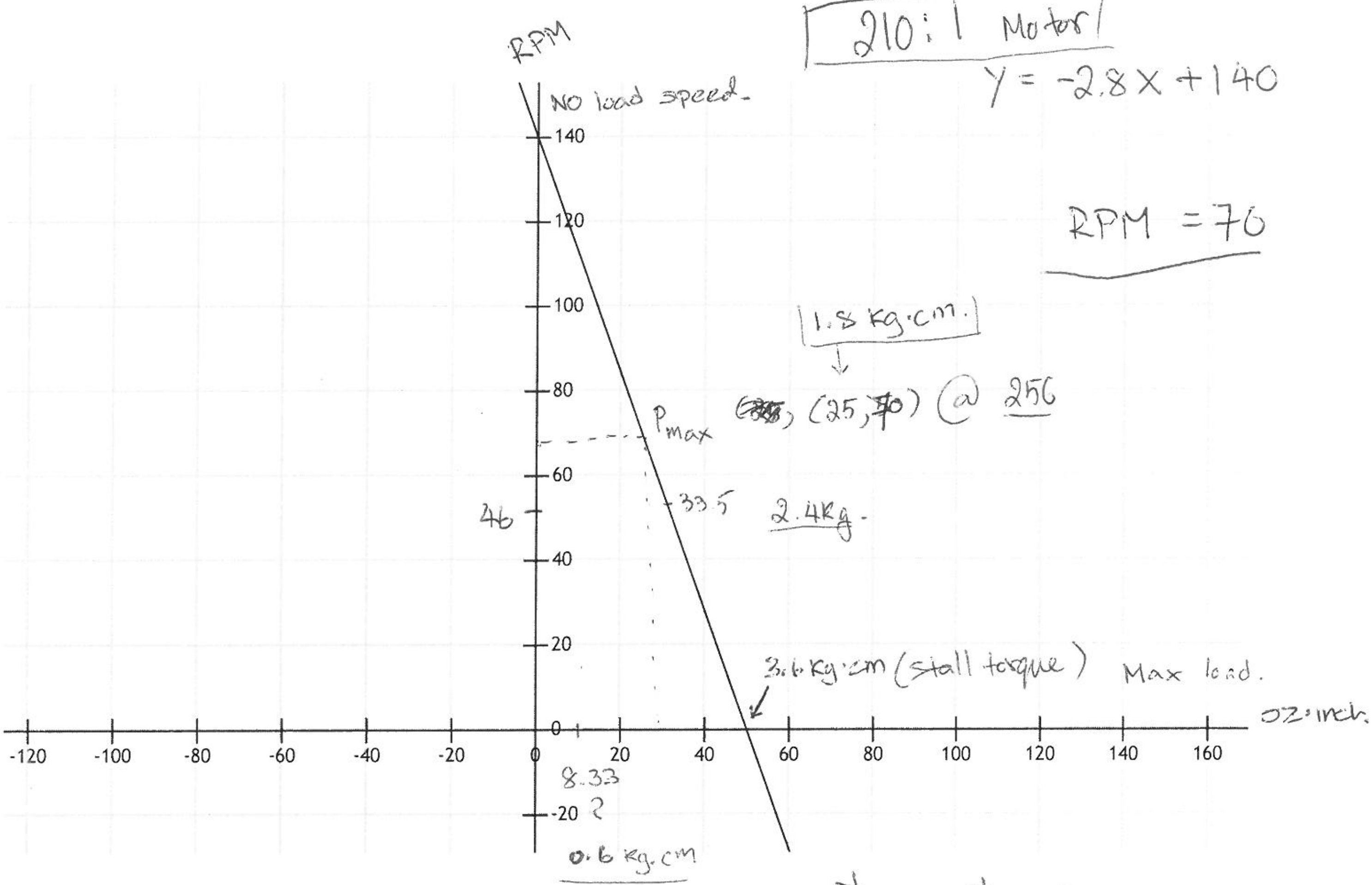
Box

ryan plan A ryan L Design
Edward plan B Edward Box ~~Change~~



- Calculate torqe produce in 210:0 Motor.
- Find ideal ~~wei~~ Net weight of the robot.

load index



= 1.2 kg

$\vec{\tau}_{Pmax} = \vec{r} \times$ Weight

$1.8 = \vec{r} \times W.$

$\frac{1.8}{2} =$ 0.9 kg. Max speed.

Net weight = 1.1 kg.

- .200kg.

ϕ = 40 mm

~~$\vec{\tau} = r \times \vec{F}_W$~~

~~$\vec{\tau}$ 0.02 × 10.78~~

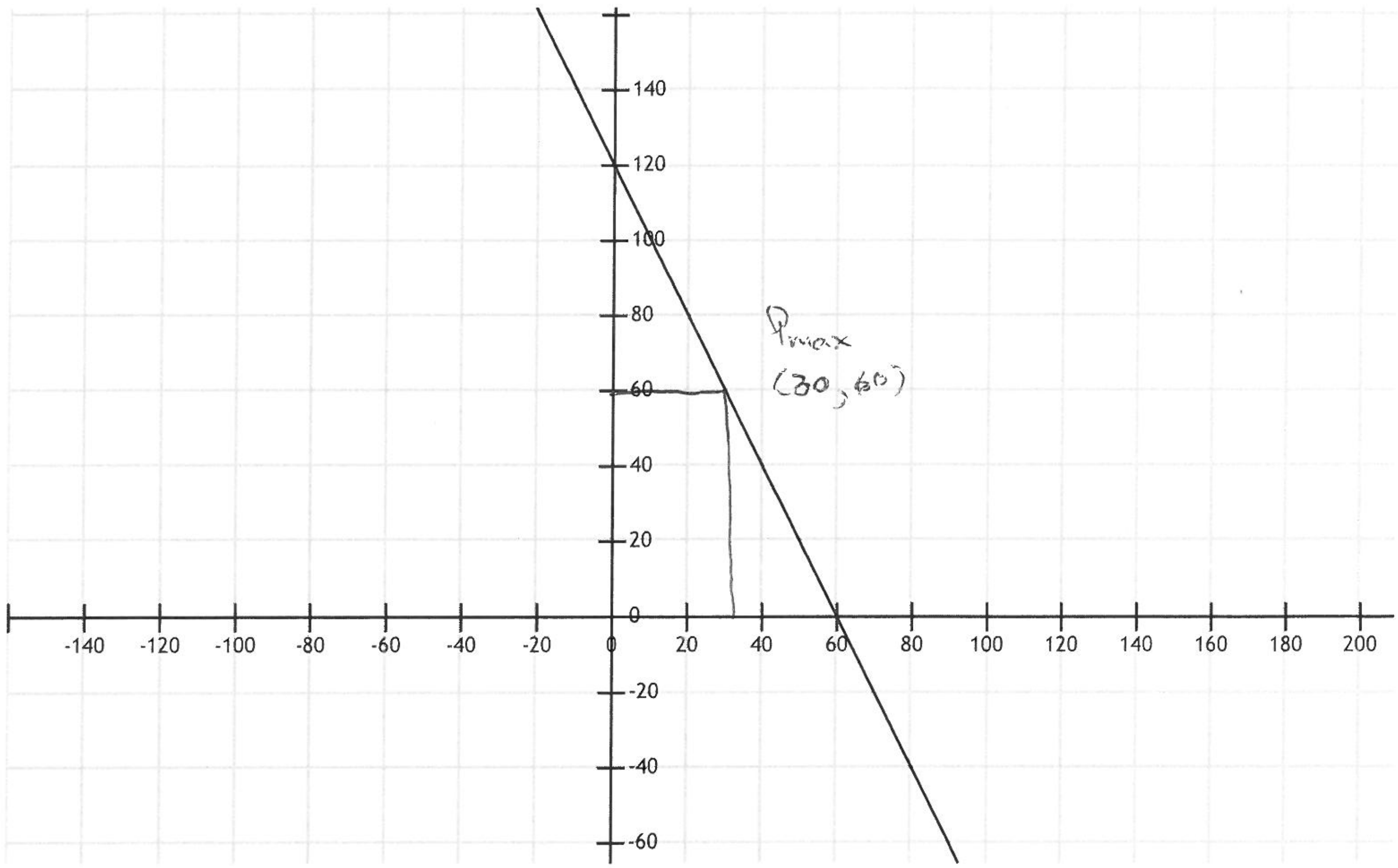
~~$\vec{\tau}$ = 0.2156 Nm.~~

~~$\vec{F} = mg$~~

~~= 1.1 × 9.8~~

~~= 10.78 N~~

250:1



r = 2cm.

$\vec{\tau} = r \times \vec{F}$

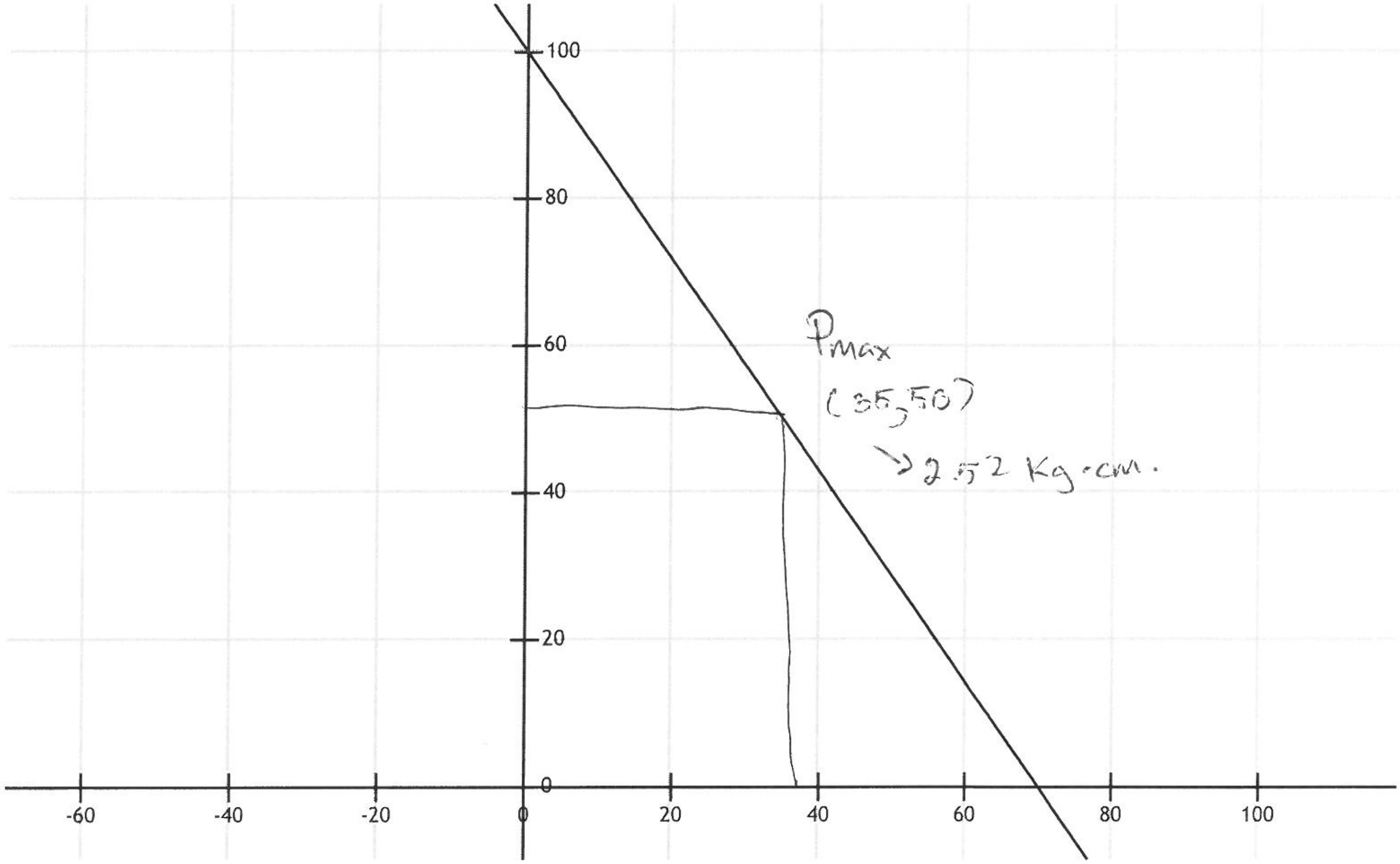
$\vec{\tau} = r \times W.$

$2.16 = 2 \times W$

$W = 1.08$ Kg.

RPM = 60

298 : 1



r = 2cm

$\vec{\tau} = r \times \vec{F}$

$\vec{\tau} = r \times W$

$2.52 = 2 \times W$

$W = 1.26$ Kg.

RPM = 50

Code Segments

Following the Line through the PID Algorithm:

```
void sensors_read() {
 sensors_average = 0;
 sensors_sum = 0;

 for (int i = 0; i < 3; ++i) {
  sensors[i] = analogRead(i);
  //Calculating the weighted mean
  sensors_average += sensors[i] * i * 1000;
  //Calculating sum of sensor readings
  sensors_sum += int(sensors[i]);
 }
}

void pid_calc() {
 Position = int(sensors_average / sensors_sum);
 proportional = Position - 744;
 derivative = proportional - last_proportional;
 last_proportional = proportional;
 error_value = (proportional * Kp + derivative * Kd);
}

void calc_turn() {
 if (error_value < -256) {
  error_value = -256;
 }

 if (error_value > 256) {
  error_value = 256;
 }

 //right turn
 if (error_value < 0) {
  left_speed = max_speed + error_value;
  right_speed = max_speed;
 }

 //left turn
 else {
  left_speed = max_speed;
  right_speed = max_speed - error_value;
 }
}
```

Code Segments

Following the Line through the PID Algorithm:

```
void sensors_read() {
 sensors_average = 0;
 sensors_sum = 0;

 for (int i = 0; i < 3; ++i) {
  sensors[i] = analogRead(i);
  //Calculating the weighted mean
  sensors_average += sensors[i] * i * 1000;
  //Calculating sum of sensor readings
  sensors_sum += int(sensors[i]);
 }
}

void pid_calc() {
 Position = int(sensors_average / sensors_sum);
 proportional = Position - 744;
 derivative = proportional - last_proportional;
 last_proportional = proportional;
 error_value = (proportional * Kp + derivative * Kd);
}

void calc_turn() {
 if (error_value < -256) {
  error_value = -256;
 }

 if (error_value > 256) {
  error_value = 256;
 }

 //right turn
 if (error_value < 0) {
  left_speed = max_speed + error_value;
  right_speed = max_speed;
 }

 //left turn
 else {
  left_speed = max_speed;
  right_speed = max_speed - error_value;
 }
}
```

Following the Wall with Analog Infrared Sensors

```
void IRwall(int lowvalue, int highvalue) {
 x = analogRead(IRside);

 if( x >= lowvalue && x <= highvalue ) {
  motor_drive1(255, 255, forward1);
 }

 if( x < lowvalue ) {
  motor_drive1(170, 70, forward1);
 }

 if( x > lowvalue ) {
  motor_drive1(70, 170, forward1);
 }

 if( x > highvalue) {
  motor_drive1(170, 70, forward1);
 }

 if( x < highvalue) {
  motor_drive1(70, 170, forward1);
 }

 if( lowvalue == highvalue) {
  motor_drive1(170, 170, forward1);
 }
}
```

Following the Wall with Analog Infrared Sensors

```
void IRwall(int lowvalue, int highvalue) {
 x = analogRead(IRside);

 if( x >= lowvalue && x <= highvalue ) {
  motor_drive1(255, 255, forward1);
 }

 if( x < lowvalue ) {
  motor_drive1(170, 70, forward1);
 }

 if( x > lowvalue ) {
  motor_drive1(70, 170, forward1);
 }

 if( x > highvalue) {
  motor_drive1(170, 70, forward1);
 }

 if( x < highvalue) {
  motor_drive1(70, 170, forward1);
 }

 if( lowvalue == highvalue) {
  motor_drive1(170, 170, forward1);
 }
}
```

Step-by-step Navigation of the Arena.

```
void navigate() {
 //TSturn('r');

 if(sensors_sum > 750 && sensors_sum < 1400) {
  PID();
 } else {
  IRwall(168, 181);
 }
}

void count0 () {
 TSturn('r');

 if(sensors_sum > 750 && sensors_sum < 1400 ) {
  PID();
 } else {
  motor_drive1(250, 250, forward1);
 }
}

void stopfordowel() {
 if(digitalRead(IRdowel) == 0) {
  motor_drive1(0,0,still1);
  delay(2000);
 }
}

void count1 () {
 stopfordowel();
 TSturn('r');
 navigate();
}

void count2() {
 TSturn('r');

 if(digitalRead(IRdowel) == 0) {
  motor_drive1(0,0,still1);
  delay(2000);
  //  IRwall(1023, 1023);
  //  delay(1000);
 } else {
  IRwall(151, 155);
 }
}
```

Step-by-step Navigation of the Arena.

```
void navigate() {
 //TSturn('r');

 if(sensors_sum > 750 && sensors_sum < 1400) {
  PID();
 } else {
  IRwall(168, 181);
 }
}

void count0 () {
 TSturn('r');

 if(sensors_sum > 750 && sensors_sum < 1400 ) {
  PID();
 } else {
  motor_drive1(250, 250, forward1);
 }
}

void stopfordowel() {
 if(digitalRead(IRdowel) == 0) {
  motor_drive1(0,0,still1);
  delay(2000);
 }
}

void count1 () {
 stopfordowel();
 TSturn('r');
 navigate();
}

void count2() {
 TSturn('r');

 if(digitalRead(IRdowel) == 0) {
  motor_drive1(0,0,still1);
  delay(2000);
  //  IRwall(1023, 1023);
  //  delay(1000);
 } else {
  IRwall(151, 155);
 }
}
```

```
void count3() {

  if (analogRead(CSright) > 250) {
   delay(150); //go forward a bit more after see black line with CSright sensor
   motor_drive1(250, 210, onewheelright1);
   delay(650); //prevent A2 sensor to see straight black line while turning
   //A2 is the left most sensor
   while (analogRead(A2) < 500) {
    delay(1);
   }
   if (count == 3) count++;
  }
  else {
   PID();
  }
 }

void count4(){
 TSturn('l');
 if (sensors_sum > 750 && sensors_sum < 1400 ) {
  PID();
 } else {
  motor_drive1(250, 250, forward1);
 }
}

void count5() {
 if (analogRead(CSleft) > 250) {
  motor_drive1(250, 0, left1);
  delay(400);
  motor_drive1(250, 250, backward1);
  delay(100);
  while (analogRead(A0) < 500) {
   delay(1);
  }
  if (count == 5) count++;
 }
 else {
  PID();
 }
}

void count6() {
 TSturn('l');
 stopfordowel();
 if (sensors_sum > 750 && sensors_sum < 1400 ) {
  PID();
 } else {
```

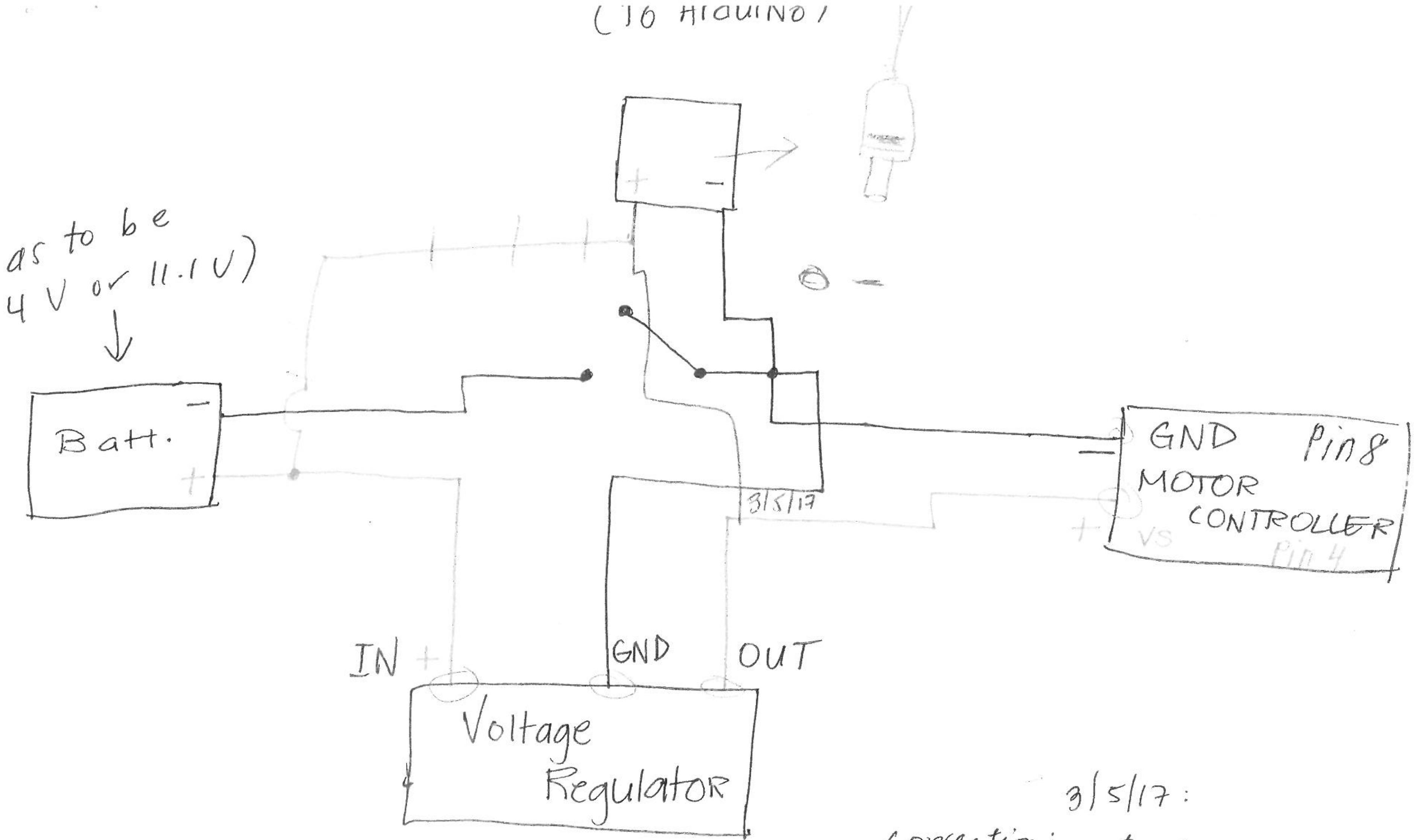
```
void count3() {

  if(analogRead(CSright) > 250) {
   delay(150); //go forward a bit more after see black line with CSright sensor
   motor_drive1(250, 210, onewheelright1);
   delay(650); //prevent A2 sensor to see straight black line while turning
   //A2 is the left most sensor
   while (analogRead(A2) < 500) {
    delay(1);
   }
   if(count == 3) count++;
  }
  else {
   PID();
  }
 }

void count4(){
 TSturn('l');
 if(sensors_sum > 750 && sensors_sum < 1400 ) {
  PID();
 } else {
  motor_drive1(250, 250, forward1);
 }
}

void count5() {
 if(analogRead(CSleft) > 250) {
  motor_drive1(250, 0, left1);
  delay(400);
  motor_drive1(250, 250, backward1);
  delay(100);
  while (analogRead(A0) < 500) {
   delay(1);
  }
  if(count == 5) count++;
 }
 else {
  PID();
 }
}

void count6() {
 TSturn('l');
 stopfordowel();
 if(sensors_sum > 750 && sensors_sum < 1400 ) {
  PID();
 } else {
```

```
    motor_drive1(200, 200, forward1);
   }
  }

  void count7() {
   TSturn('l');

   if (sensors_sum > 750 && sensors_sum < 1400 ) {
    PID();
   } else {
    motor_drive1(200, 200, forward1);
   }
  }

  void count8() {
   TSturn('l');
   if (sensors_sum > 750 && sensors_sum < 1400 ) {
    PID();
   } else {
    motor_drive1(200, 200, forward1);
   }
  }

 void count9() {
  stopfordowel();
  if (analogRead(CSleft) > 250) {
   delay(550);
   motor_drive1(250, 250, left1);
   delay(650);
   while (analogRead(A0) < 500) {
    delay(1);
   }
   if (count == 9) count++;
  }
  else if (sensors_sum < 1400 ) {
   PID();
  }
 }

void count10() {
 TSturn('r');
 stopfordowel();
 if (sensors_sum > 750 && sensors_sum < 1400 ) {
  PID();
 } else {
  motor_drive1(200, 200, forward1);
 }
}
```

```
    motor_drive1(200, 200, forward1);
   }
  }

  void count7() {
   TSturn('l');

   if (sensors_sum > 750 && sensors_sum < 1400 ) {
    PID();
   } else {
    motor_drive1(200, 200, forward1);
   }
  }

  void count8() {
   TSturn('l');
   if (sensors_sum > 750 && sensors_sum < 1400 ) {
    PID();
   } else {
    motor_drive1(200, 200, forward1);
   }
  }

  void count9() {
   stopfordowel();
   if (analogRead(CSleft) > 250) {
    delay(550);
    motor_drive1(250, 250, left1);
    delay(650);
    while (analogRead(A0) < 500) {
     delay(1);
    }
    if (count == 9) count++;
   }
   else if (sensors_sum < 1400 ) {
    PID();
   }
  }

 void count10() {
  TSturn('r');
  stopfordowel();
  if (sensors_sum > 750 && sensors_sum < 1400 ) {
   PID();
  } else {
   motor_drive1(200, 200, forward1);
  }
 }
```

```
void count11() {
  TSturn('r');
  stopfordowel();
  navigate();
}

void count12() {
 TSturn('p');
 PID();
 motor_drive1(0,0,still1);
 delay(5000);

}
```

```
void count11() {
  TSturn('r');
  stopfordowel();
  navigate();
}

void count12() {
  TSturn('p');
  PID();
  motor_drive1(0,0,still1);
  delay(5000);

}
```



3/5/17:

Correction: Arduino should be from Voltage Regulator Out (not In)

H-Bridge L298N on "Sparkfun" PCB board

PIN DESCRIPTION:

PINS	DESCRIPTION
OUT1	Output connected directly to motor 1
VS	Voltage Supply for Motors (2.5V to 46V MAX)
ENA	Enable Motor 1 (ON/OFF) or can be used to control Input Voltage through PWM for the Motor 1
GND	Ground for the H-Bridge
CSA	Current Sensing for Motor 1. Connected to the GROUND or Current Sensor chip.
OUT2	Output connected directly to motor 1
IN1	Digital Input for Motor 1 (HIGH/LOW) use to specify polarity on the Motor 1
IN2	Digital Input for Motor 1 (HIGH/LOW) use to specify polarity on the Motor 1
VLS	Voltage Supply for H-Bridge (4.5V to 7V Max)
ENB	Enable Motor 2 (ON/OFF) or can be used to control Input Voltage through PWM for the Motor 2
OUT3	Output connected directly to motor 2
CSB	Current Sensing for Motor 2. Connected to the GROUND or Current Sensor chip.
IN3	Digital Input for Motor 2 (HIGH/LOW) use to specify polarity on the Motor 2
IN4	Digital Input for Motor 2 (HIGH/LOW) use to specify polarity on the Motor 2
OUT4	Output connected directly to motor 2

MOTOR CONNECTION:

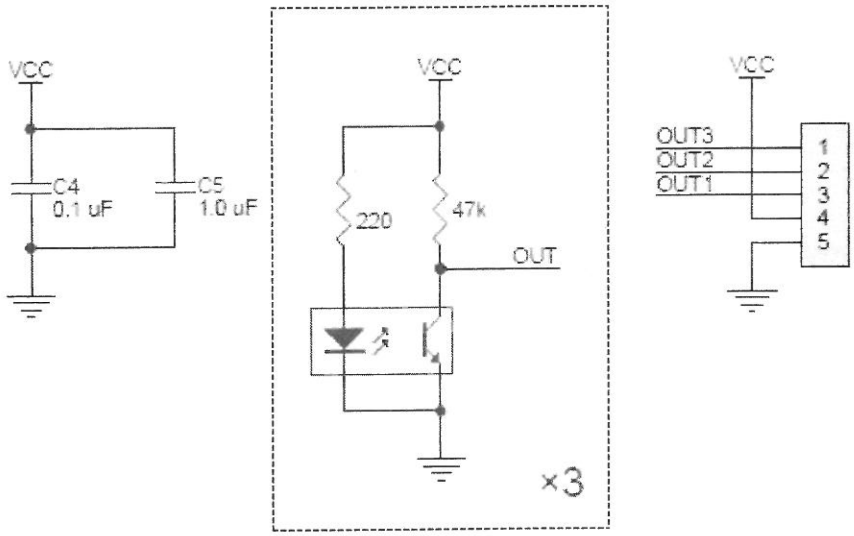
MOTOR 1	
PINS	CONNECTED
IN1, IN2	BOARD (Digital Pin)
OUT1, OUT2	MOTOR
ENA	BOARD (Digital Pin / PWM Pin)
CSA	GROUND / Current Sensor

MOTOR 2	
PINS	CONNECTED
IN3, IN4	BOARD (Digital Pin)
OUT3, OUT4	MOTOR
ENB	BOARD (Digital Pin / PWM Pin)
CSB	GROUND / Current Sensor

INPUT LOGIC:

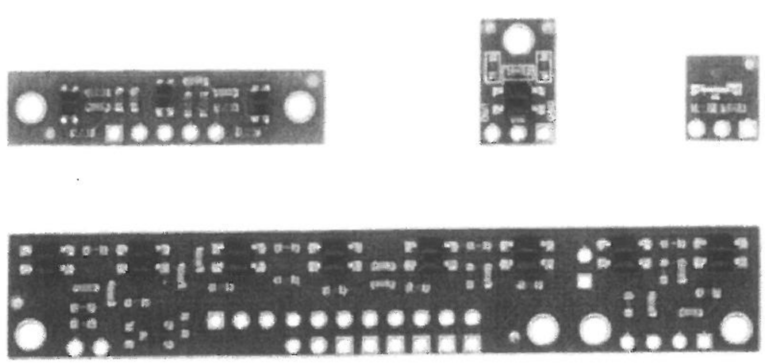
INPUT 1 or 3	INPUT 2 or 4	Description
LOW	HIGH	Motors move one Direction
HIGH	LOW	Motors move Opposite Direction
LOW	LOW	Motors have no Voltage, free rotation
HIGH	HIGH	Unpredictable Statement

NOTE: You can Connect INPUT pins of H-Bridge directly to Ground and VLS power supply if motor will be used to rotate only single direction.



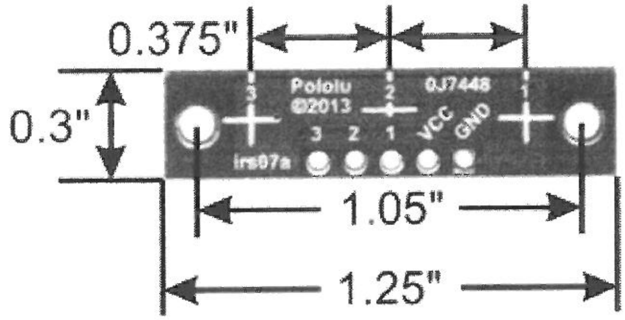
This schematic is also available as a downloadable pdf (115k pdf).

For an alternative array with eight sensors and the ability to turn off the IR LEDs to limit power consumption, consider our QTR-8A reflectance sensor array. For individual reflectance sensors, consider our QTR-1A and QTR-L-1A.



QTR sensor size comparison. Clockwise from top left: QTR-3RC, QTR-1RC, QTR-L-1RC, QTR-8RC.

Specifications

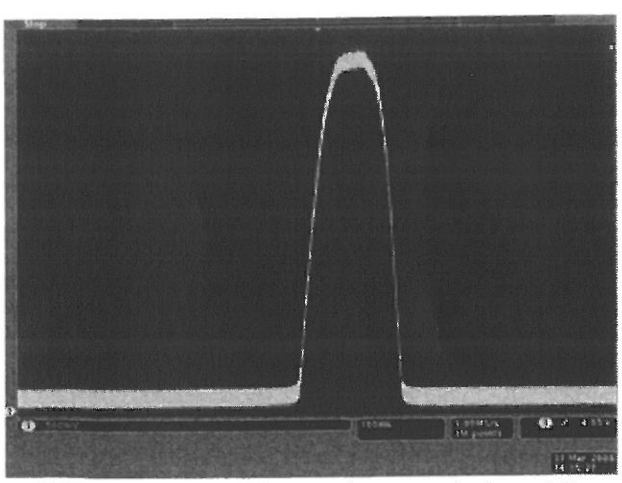


- Dimensions: 1.25" × 0.3" × 0.1" (32 × 8 × 3 mm) (without header pins installed)
- Operating voltage: 5.0 V
- Supply current: 50 mA
- Output format: 3 analog voltages
- Output voltage range: 0 V to supplied voltage
- Optimal sensing distance: 0.125" (3 mm)
- Maximum recommended sensing distance: 0.25" (6 mm)
- Weight without header pins: 0.02 oz (0.6 g)

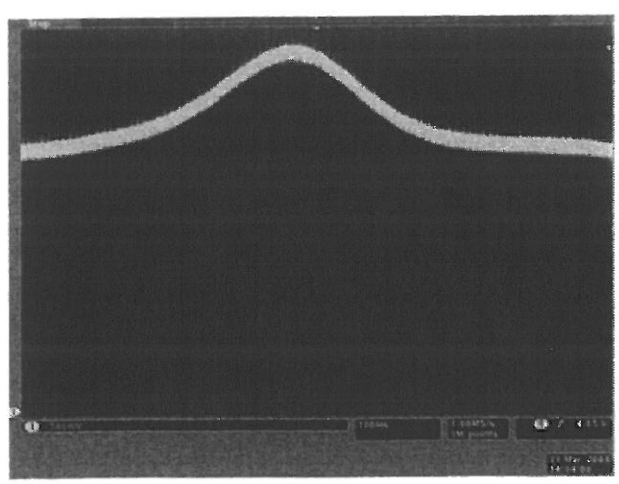
Interfacing with the QTR-3A Outputs
There are several ways you can interface with the QTR-3A outputs:

- Use a microcontroller's analog-to-digital converter (ADC) to measure the voltages.
- Use a comparator with an adjustable threshold to convert each analog voltage into a digital (i.e. black/white) signal that can be read by the digital I/O line of a microcontroller.
- Connect each output directly to a digital I/O line of a microcontroller and rely upon its internal comparator.

This last method will work if you are able to get high reflectance from your white surface as depicted in the left image, but will probably fail if you have a lower-reflectance signal profile like the one on the right. (Please note that these images show the output of a QTR-1A, which uses a sensor with slightly different characteristics than the ones on the QTR-3A.)

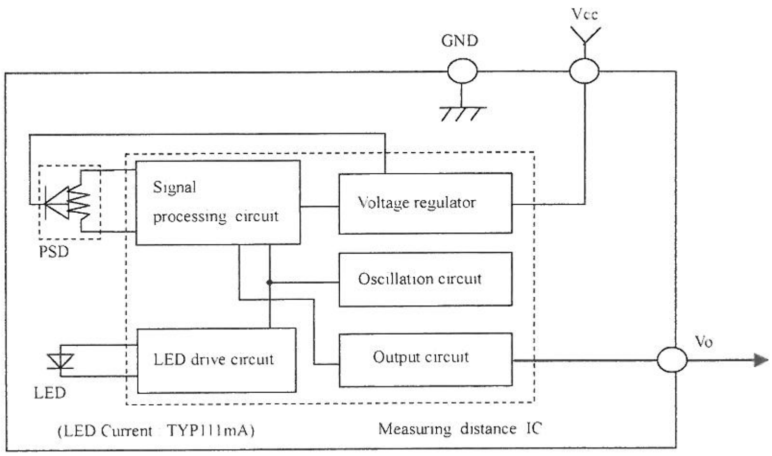


QTR-1A output 1/8" away from a spinning white disk with a black line on it.

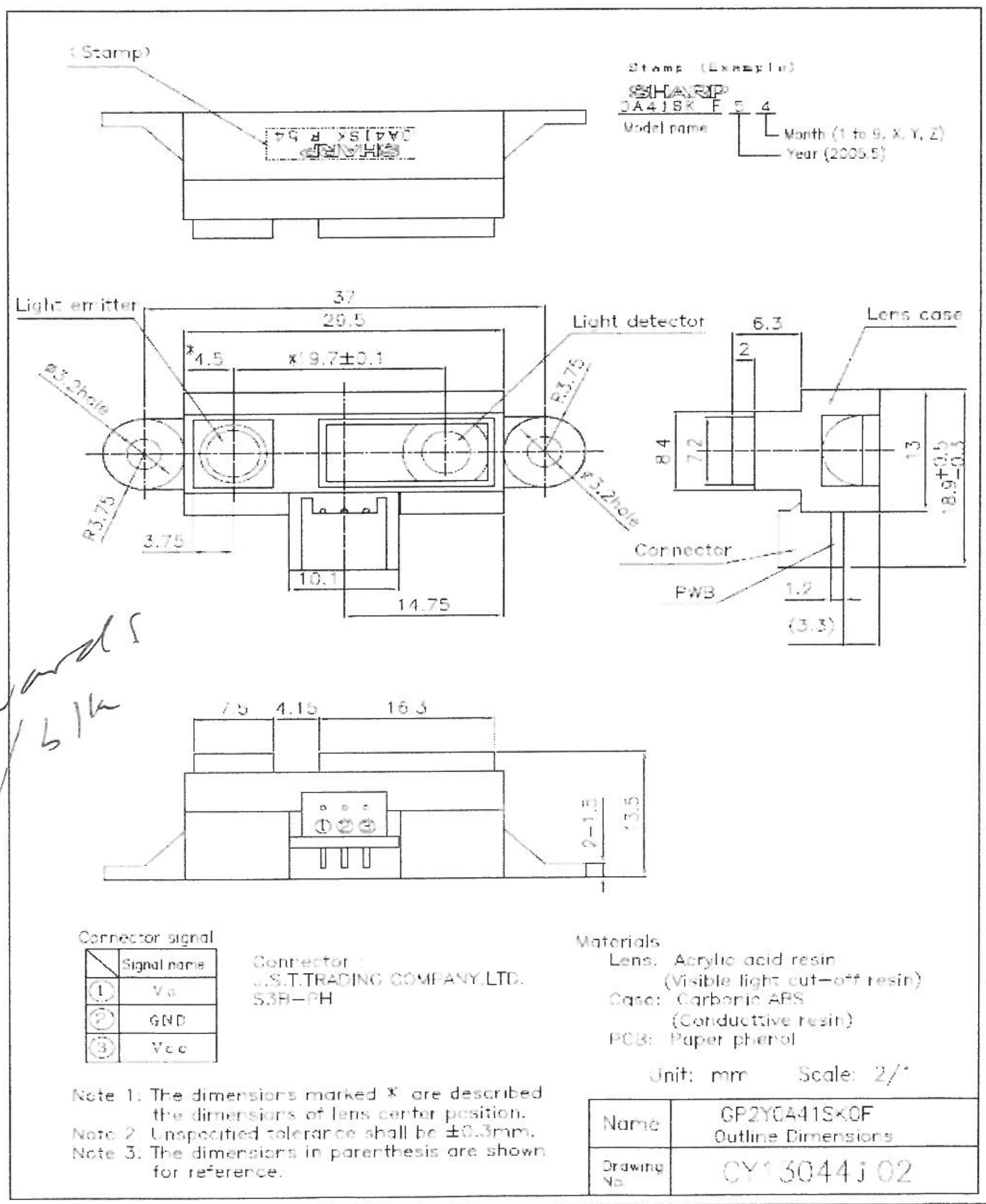


QTR-1A output 3/8" away from a spinning white disk with a black line on it.

■Schematic



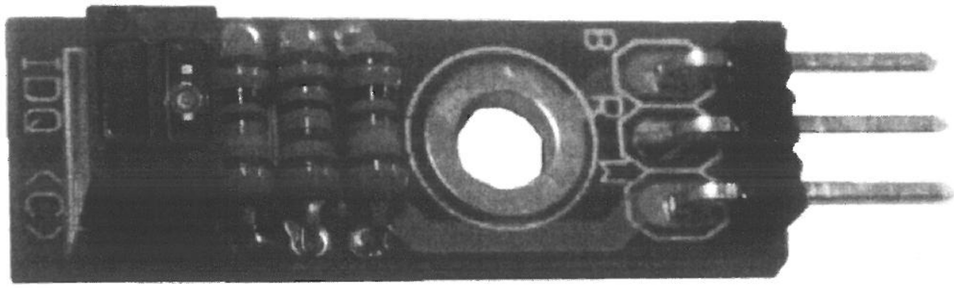
■Outline



backwards
red / blk

Detailed Pin Description

The figure below shows the pins arrangement of the CS491000, and the following table shown the Detailed Pin Description. Note the pins at the connector denoted by the letters B, R, and W.



IDQ Contrast Sensor Pin-Out Connectivity

Pin	*Symbol*	*Description*
1	B	GND Ground pin.
2	R	Output pin. Connect to any C Stamp I/O pin for digital input into the C Stamp or to an Analog-to-Digital Converter input for analog input into the C Stamp. The lower the voltage at this pin, the lighter the surface below the sensor is. Conversely, the higher the voltage at this pin, the darker the surface below the sensor is.
3	W	VDD Supply 4.2 – 5.5 V. positive