

# Carrom Strike 1.6

## The Syndicate

An autonomous carrom playing bot, that can play against human opponents!

### Topic List:

1. [Final Results/Home \(Video links etc\)](#)
2. [Introduction](#)
3. [Basics/Concepts](#)
4. [Timeline/Plan of Action](#)
5. [Theory and Details](#)
  - a. [A4988 - Stepper Motor Driver](#)
  - b. [Image Processing](#)
  - c. [Artificial Intelligence](#)
  - d. [Arduino](#)
  - e. [Mechanical Structure](#)
6. [Pictures](#)
7. [Components \(brief list\)](#)
8. [References](#)
9. [Team](#)

## FINAL RESULTS/HOME

[Working Video](#)

[Image Processing Code](#)

[Algorithm](#)

[Full Code\(IP and Algorithm\)](#)

[Arduino Code](#)

## INTRODUCTION

Still think you need atleast 2 people to play a simple game of carrom? Have a look at Carrom Strike 1.6! It is an autonomous bot that plays carrom against human opponents. With an attached Webcam for situational analysis, an Image Processing algorithm for coin detection and identification, a decision making algorithm for shot selection, and the mechanical/electrical structure for taking shots, CS1.6 is fully equipped to play a basic game of carrom with you, and defeat you unmistakably.

## **BASIC CONCEPTS**

The following areas play a major part in development of the bot:

1. Image Processing (Coin detection and identification)
2. Decision Making Algorithm (Shot possibilities and decision)
3. Mechanical Structure (Movement, Aiming and Shooting)
4. Microcontrollers and Drivers (Motor Co-ordination and Control)

## **REFERENCES:**

### **OpenCV(Python) for Image Processing**

<http://docs.opencv.org/2.4/modules/core/doc/intro.html>

Mostly the basics, and Hough Transform for Circle detection.

### **Microcontrollers(Arduino)**

<https://stab-iitb.org/electronics-club/tutorials/arduino/>

### **Stepper Motor Drivers**

<https://www.pololu.com/product/1182>

<https://www.pololu.com/product/1182/faqs>

### **Stepper Wire Colour Codes:**

<https://drive.google.com/open?id=0B5y8KVZJa-eCNUViTjlsZ1ZtWTQ>

## **TIMELINE/PLAN OF ACTION**

### **May 21st - May 31st:**

- Full Research work
- Decisions regarding Mechanical Structure
- Learning and developing Image Processing code  
(Refer IP code link)

### **June 1st - June 10th:**

- Planned and finished the Shot decision algorithm
- Integrated Image Processing and Shot Algorithm
- Completed Arduino Code for Motor Control using Serial Communication(UART)  
(Refer all Code links)

### **June 11th - June 20th:**

- Made primary mechanical assembly
- Achieved a basic assembly movement using motors(Linear Position and Angle Setting)  
(Intermediate Stage)

### **June 21st - June 27th:**

- Finalised the mechanical structure and code format to suit final game format
- Refer the final video on the link above!

## **THEORY AND DETAILS**

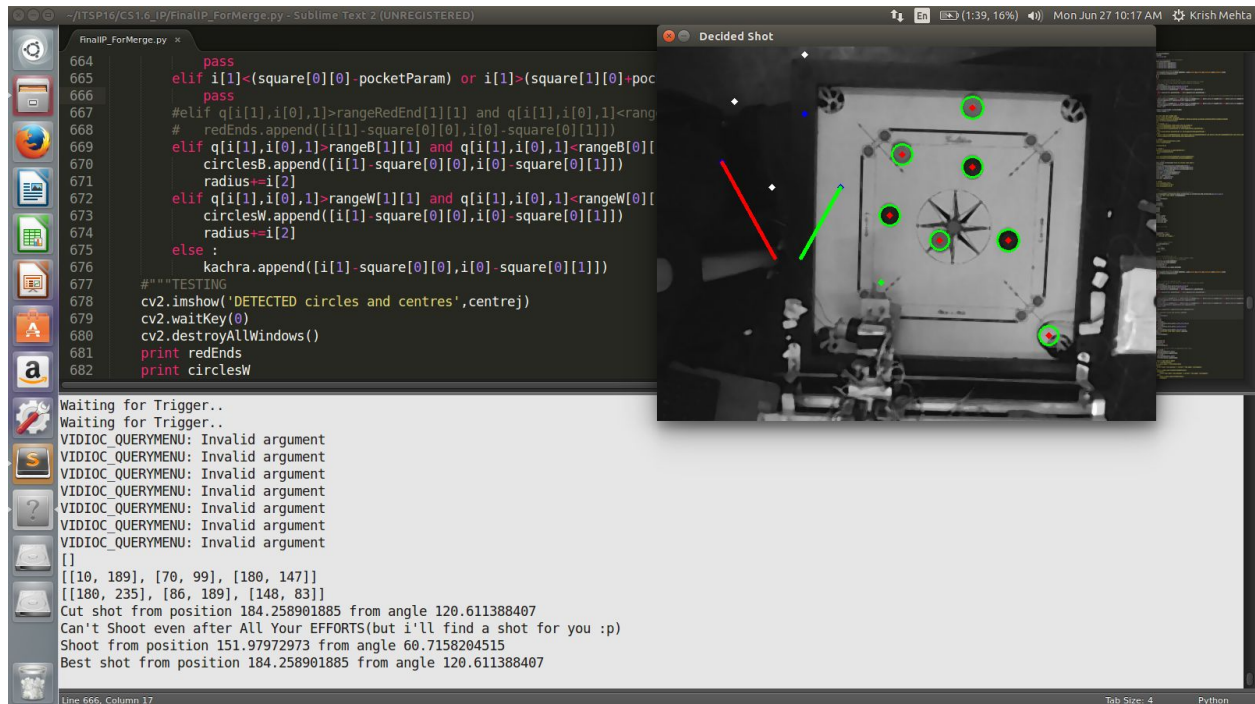
### **Image Processing:**

The only thing we needed was detections of circles. Subsequently, they could be easily classified based on the colour values on their centres, whether they fall in the White range or Black range.

For detection, we used a function available in OpenCV(Python) which detects circles using the "Hough Transform". In a gist, it first applies a median blur filter on the image, calculates the gradient of the image, does the thresholding and then identifies circles that fall under the criterion specified by certain parameters passed to it. Mainly we calibrated using Minimum Radius, Maximum Radius and Minimum distance between centres of two circles. Further we also narrowed down using two parameters which, qualitatively, defines what range of difference between values of pixels "in the circle" and "outside the circles" do we accept. It is like saying, how clear should be the circles you want to be counted.

Now, the colour ranges for classifying the coins as Black and White were obtained by calibration which is done once in a game at the beginning, where the user is asked to click on the centres of as many black and white coins he/she sees each color separately. Using the collected pixel values, a range is defined for each color. Apart from that, the coordinates of the pockets and the striker line are also taken for the dimensions/boundary conditions and the striker line parameters for shot decision respectively.

These are the results we achieved:



The result shown in the above picture also includes the Shot Decision algorithm. Explanation: The coins are detected and shown with red coloured centres and green circles. Their centres are transformed to the origin(hence appear shifted) and then classified as BLUE dots for Black, WHITE dots for White and GREEN dots for None/Unclassified/Out-of-the-board. Further, the possible shots are shown in red lines, you can physically verify the results and get a feel of the correctness of the shots enlisted. Eventually the GREEN LINE shows the best decided shot.

### Shot Decision Algorithm:

Once we get the centres of all the black, white & red coins on the carrom board using image processing, our bot needs to “think” and decide which shot it has to take. So, the code (written in python, it being the easiest functional programming language having a support for easy image processing) for decision making (the brain of our bot), basically maps out the centres of all coins with the pockets on the carrom board. It then checks for the possible direct shots, intermediate shots, reflection shots and cut shots, in the priority order as stated. These should cover almost 95%(not a statistically determined figure) of the scenarios arising in a typical carrom game, and thus our carrom bot has a decent accuracy to boast of.

If the shot is possible, the program returns three parameters: the x-coordinate of the striker on the striking line, the slope/angle at which the striker will be struck, and also the power/force with which it will be struck.

These three variables will be sent to the arduino IDE code so that it can give appropriate signals to the stepper motor for their movement.

CODE: [Decision Making Algorithm](#)

The PICTURES Section vividly captures the interfacing of the python code with the actual images taken from our webcam.

### Arduino Code for Motor Control:

The controlling of Motor was simple once we knew how to use A4988 Stepper Driver. So the main part of the arduino code was the serial communication with our python code (UART Protocol).

The pattern we chose was this:

On Pressing the reset button, the code passes through the setup phase where '1' is sent by Serial to the computer(USB). On the computer-side, the python code has passed through the calibration phase(one-time) and is now waiting for a Serial '1'. On receiving '1', the code executes one iteration of image-capture, coin detection-classification, shot decision and Serial communication of the parameters:

- 1.Linear Distance to be moved
- 2.Angle to be rotated
- 3.Power to be used for shot

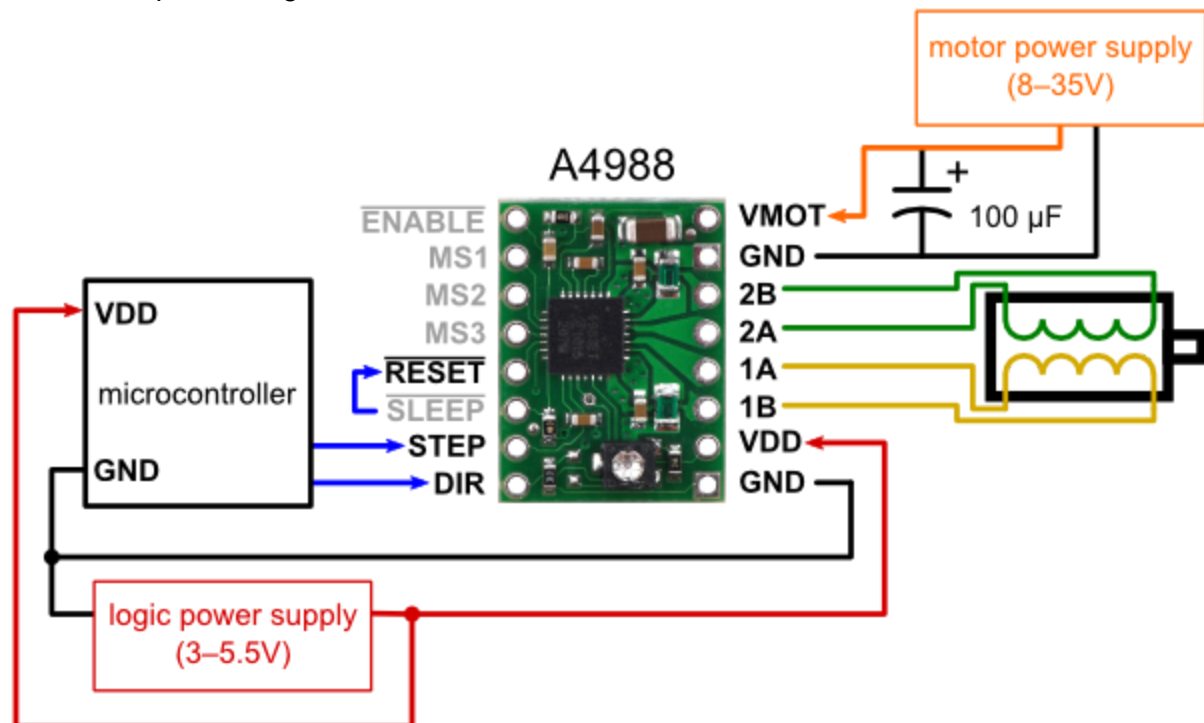
Once again , in the 'loop' of the Arduino code, it is waiting for a serial input and when it is received, the motors are instructed to move accordingly, by using the drivers.

[Here](#) is the code for Motor Control.

### Stepper Drivers:

We used the economical yet efficient enough A4988 Stepper Motor Driver.

Refer to its pinout diagram here:



The driver takes 2 logic inputs from the microcontroller, namely STEP and DIR(Direction), and obviously the VCC and GND of the microcontroller. DIR pin being HIGH/LOW determines one of

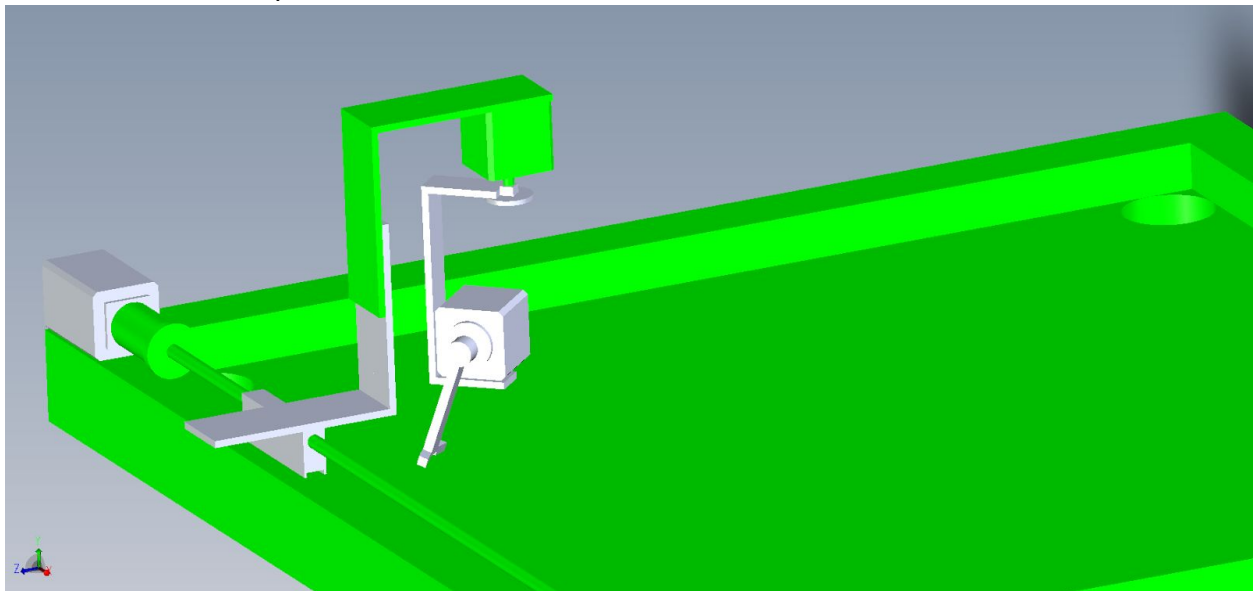
the the two directions in which the stepper can take its steps(specific minimum degree of rotation). On the STEP pin, a HIGH step will tell the stepper to take one step in the direction specified by DIR. So we simply give a Square wave from the Arduino on the STEP pin, and achieve different RPMs by changing the time period of the Square wave.

In case of 6 wired Steppers, there are 2 additional wires for centre-tapping. They are to be left open for our purpose. Which colours generally correspond to which ends of the coil can be seen in the picture in the references section.

### **Mechanical Structure:**

The mechanical structure was designed so as to be light(for easier movement) and hurdle free(for shot space). We first developed a SolidWorks model and using it as an ideal guideline, tried to perfect our mechanical structure.

Here is what we had planned, and later achieved:



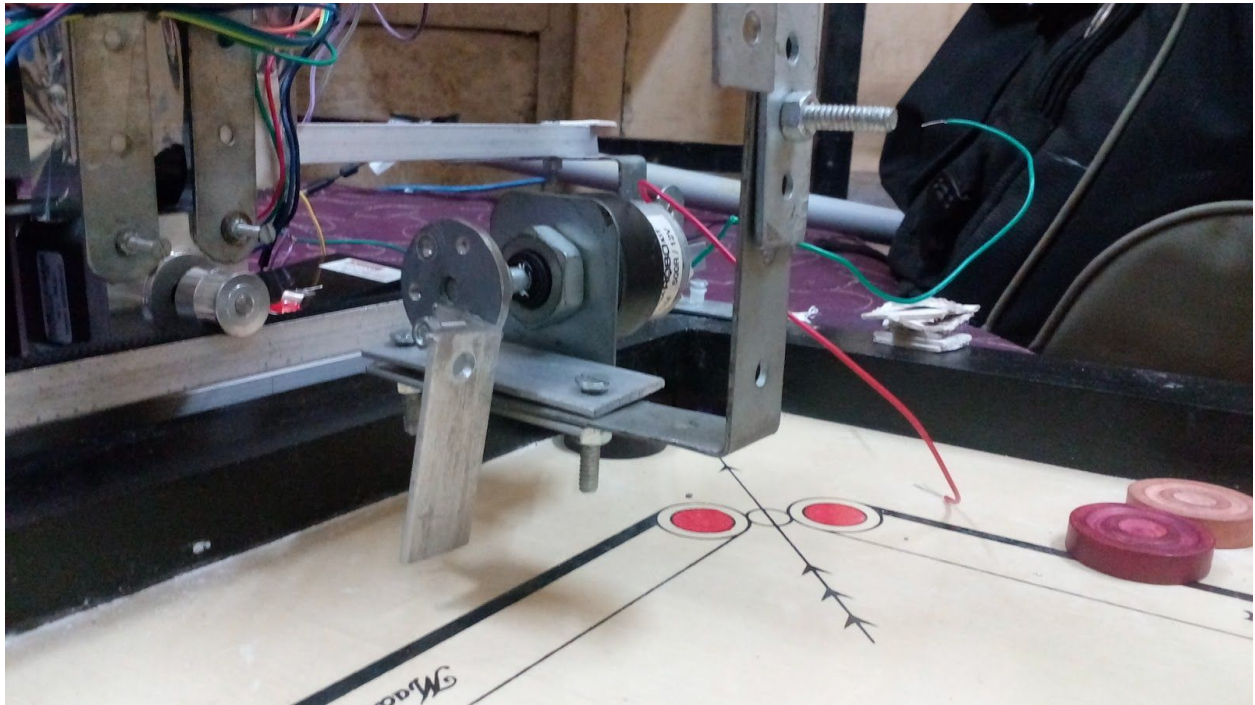
**NOTE:** The Leadscrew mechanism for linear movement was replaced by a Rack and Pinion mechanism in the final bot, because the Leadscrews of pitches around 1-2mm proved to be too slow(taking around 2-3 mins for a shot) while on increasing the pitch, larger torque was needed from the motor)

Have a look at the final look of the bot in the Pictures section.

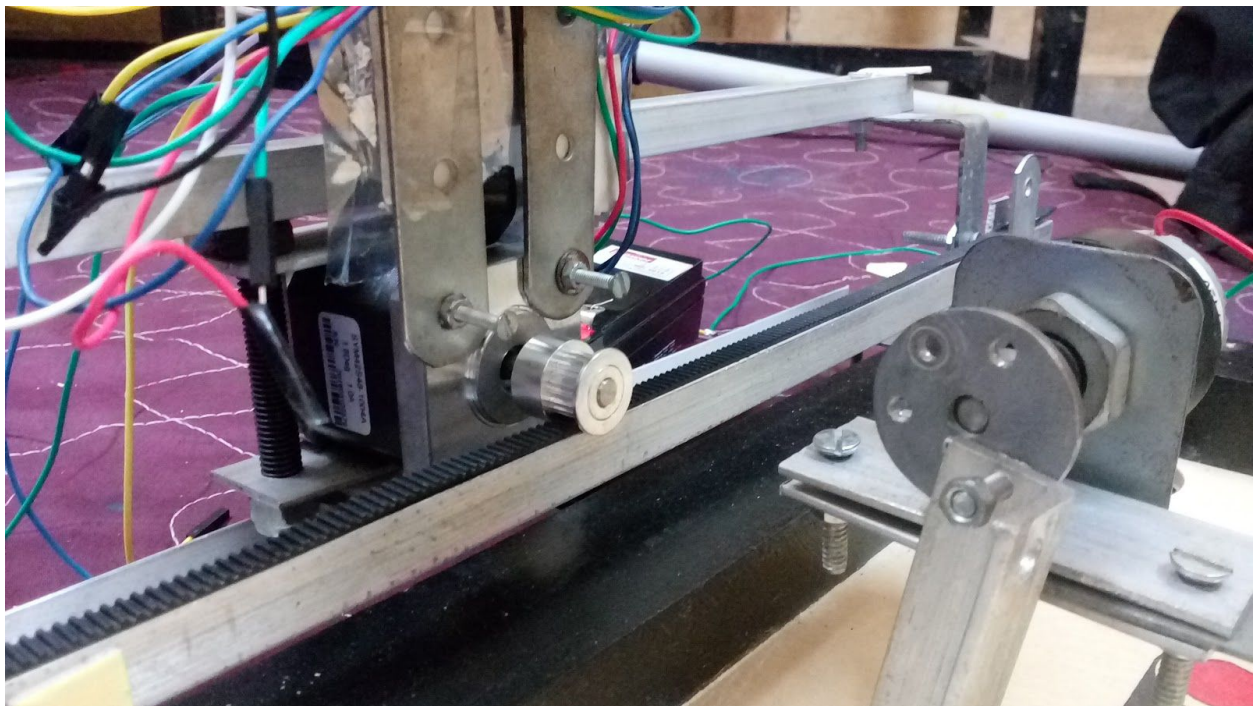


## PICTURES

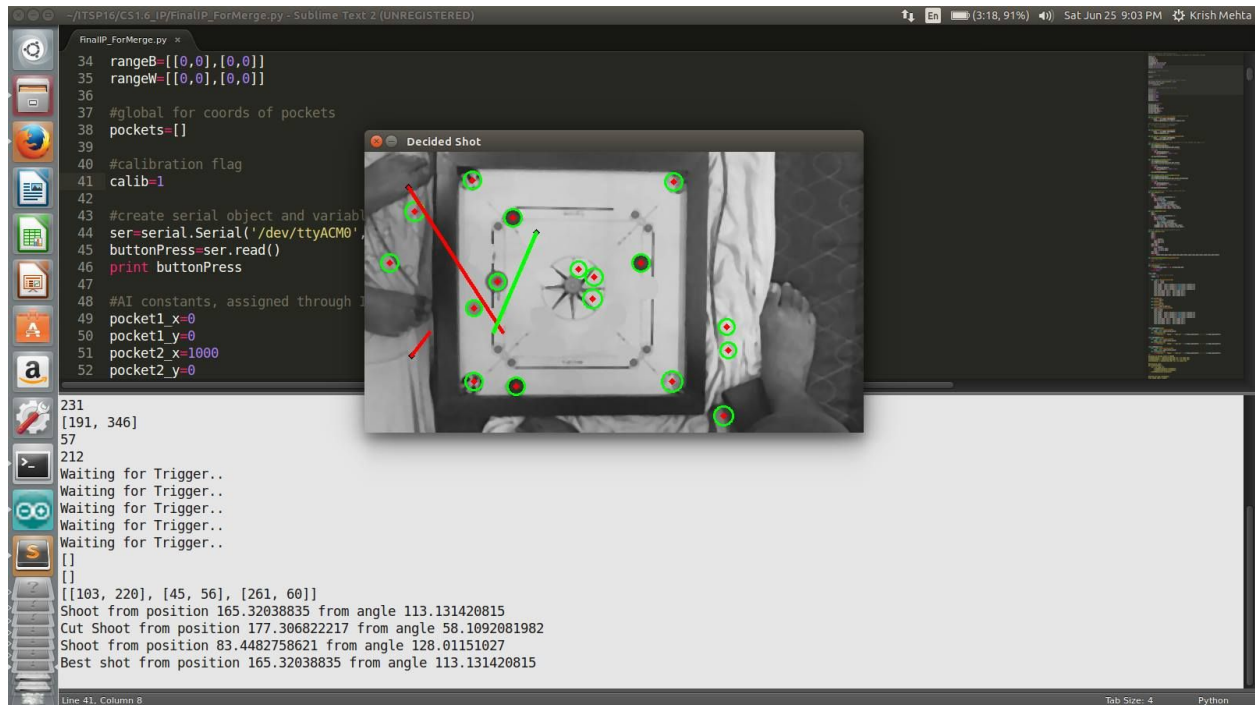
### Shooting Part



### Rack and Pinion Mechanism for Linear Motion



## Code Screenshot, showing sequencing and results of IP and Shot Decision



The screenshot shows a Sublime Text editor window titled "FinalIP\_ForMerge.py" with the following Python code:

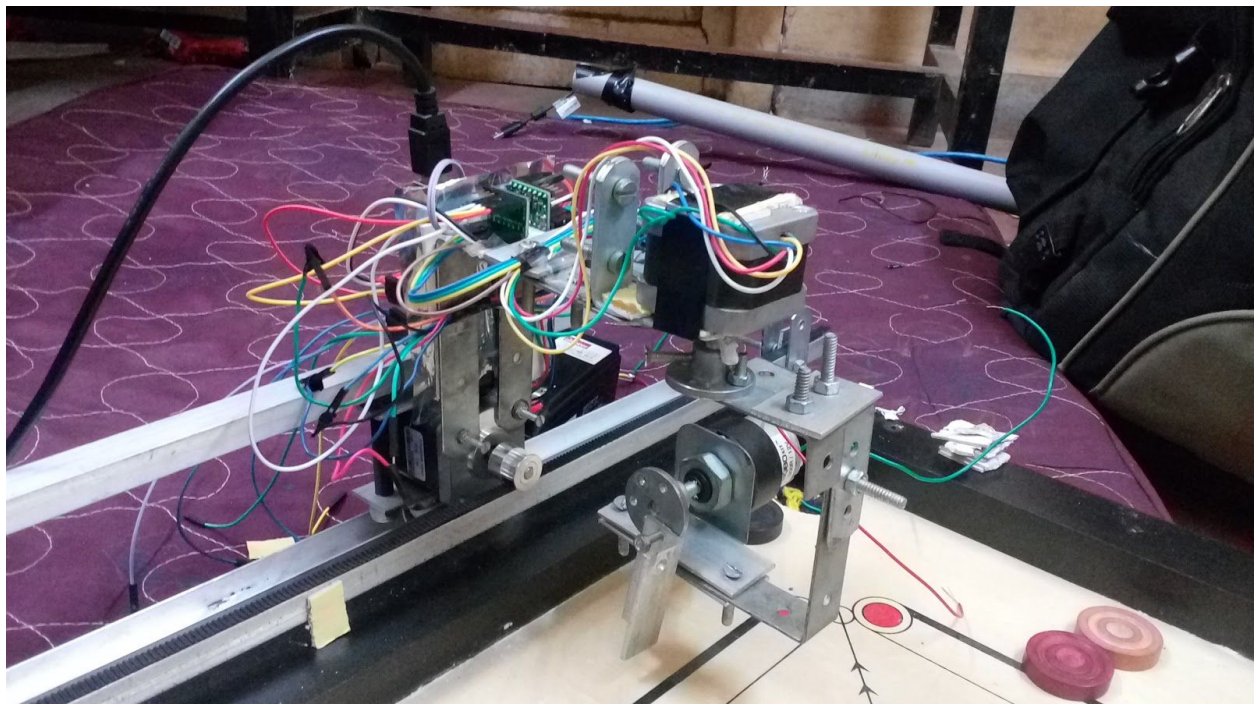
```
34 rangeB=[[0,0],[0,0]]
35 rangeW=[[0,0],[0,0]]
36
37 #global for coords of pockets
38 pockets=[]
39
40 #calibration flag
41 calib=1
42
43 #create serial object and variable
44 ser=serial.Serial('/dev/ttyACM0',
45 buttonPress=ser.read()
46 print buttonPress
47
48 #AI constants, assigned through
49 pocket1_x=0
50 pocket1_y=0
51 pocket2_x=1000
52 pocket2_y=0
```

The console output shows the following sequence of events:

```
231 [191, 346]
57
212
Waiting for Trigger..
Waiting for Trigger..
Waiting for Trigger..
Waiting for Trigger..
Waiting for Trigger..
[]
[]
[[103, 220], [45, 56], [261, 60]]
Shoot from position 165.32038835 from angle 113.131420815
Cut Shoot from position 177.306822217 from angle 58.1092081902
Shoot from position 83.4482758621 from angle 128.01151027
Best shot from position 165.32038835 from angle 113.131420815
```

A window titled "Decided Shot" is overlaid on the code, showing a pool table with a cue ball and several balls. A red line indicates the cue ball's path, and a green line indicates the target ball's path. The cue ball is positioned near the top left, and the target ball is near the bottom right.

## Final Structure





## COMPONENTS

- Stepper Motors and Drivers(Rs.2876) from Bombay Electronics, Lamington Road.
  - Mechanical Parts for construction from Mangaldeep(sum up to around Rs.1273).
  - Webcam(Rs.960) from Amazon
  - Carrom Board and coins(Rs.750) from Kid's Care, Galleria.
  - Arduino(Rs.600) from Mangaldeep.
  - Rack and Pinion(Rs.380) from Bombay Electronics, Lamington Road.
- Total Project cost: Rs. 6839.**

## The Syndicate

**Krish Mehta**

[krishmehta97@gmail.com](mailto:krishmehta97@gmail.com)

**Vrihad Agarwal**

[vrihadagarwal@gmail.com](mailto:vrihadagarwal@gmail.com)

**Archit Gupta**

[archit1197@gmail.com](mailto:archit1197@gmail.com)

**Bharat Khandelwal**

[bharatseema@gmail.com](mailto:bharatseema@gmail.com)