

# Introduction to ABB Labs

---

TA's:

Ryan Mocadlo ([mocad@wpi.edu](mailto:mocad@wpi.edu))

Adam Gatehouse ([ajgatehouse@wpi.edu](mailto:ajgatehouse@wpi.edu))

# Labs

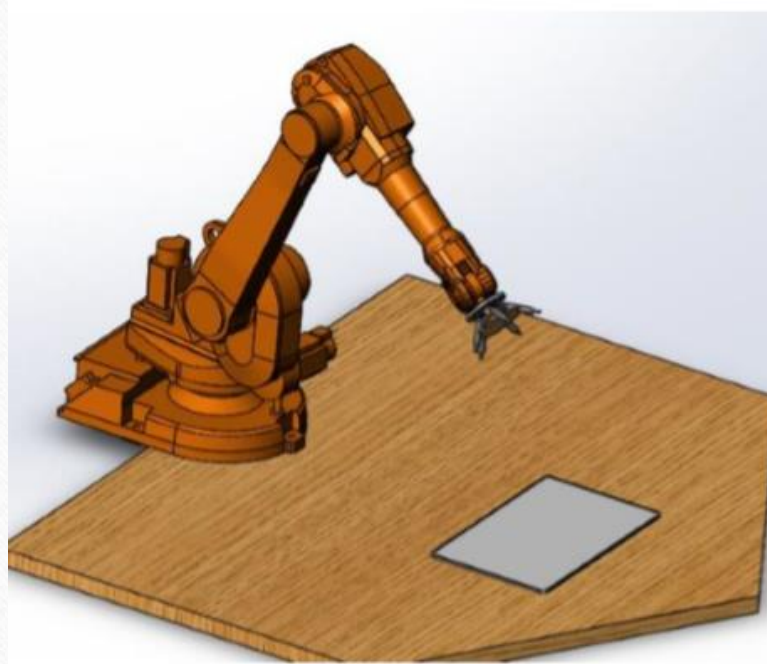
---

- In-depth lab guidelines found on Canvas
  - Must read before coming to lab section
- Total of 4 Labs:
  - Lab 1: Jogging the Robot (Getting familiar with the Pendant)
  - Lab 2: Online Robot Programming (Intro to Rapid Code)
  - Lab 3: Offline Robot Programming (Use of Robot Studio)
  - Lab 4: Palletizing Exercise (Use of external devices)



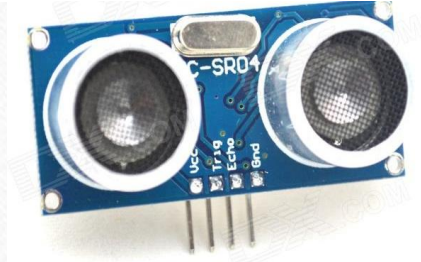
# Hardware

- ABB Robot – IRB 1600
  - 6 axis
  - Handling Capacity: 6kg
  - Reach : 1.45m
  - I/O Device
- Flex Pendant
- PLC



# External Hardware

- Other peripheral devices may be used with the ABB
  - Cameras
  - Force Sensors
  - Limit Switches
- Communication with ABB
  - I/O
  - PLC
  - TCP/IP



# EOAT



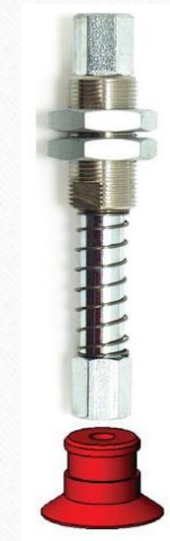
Parallel Gripper



Angular Gripper



Radial Gripper



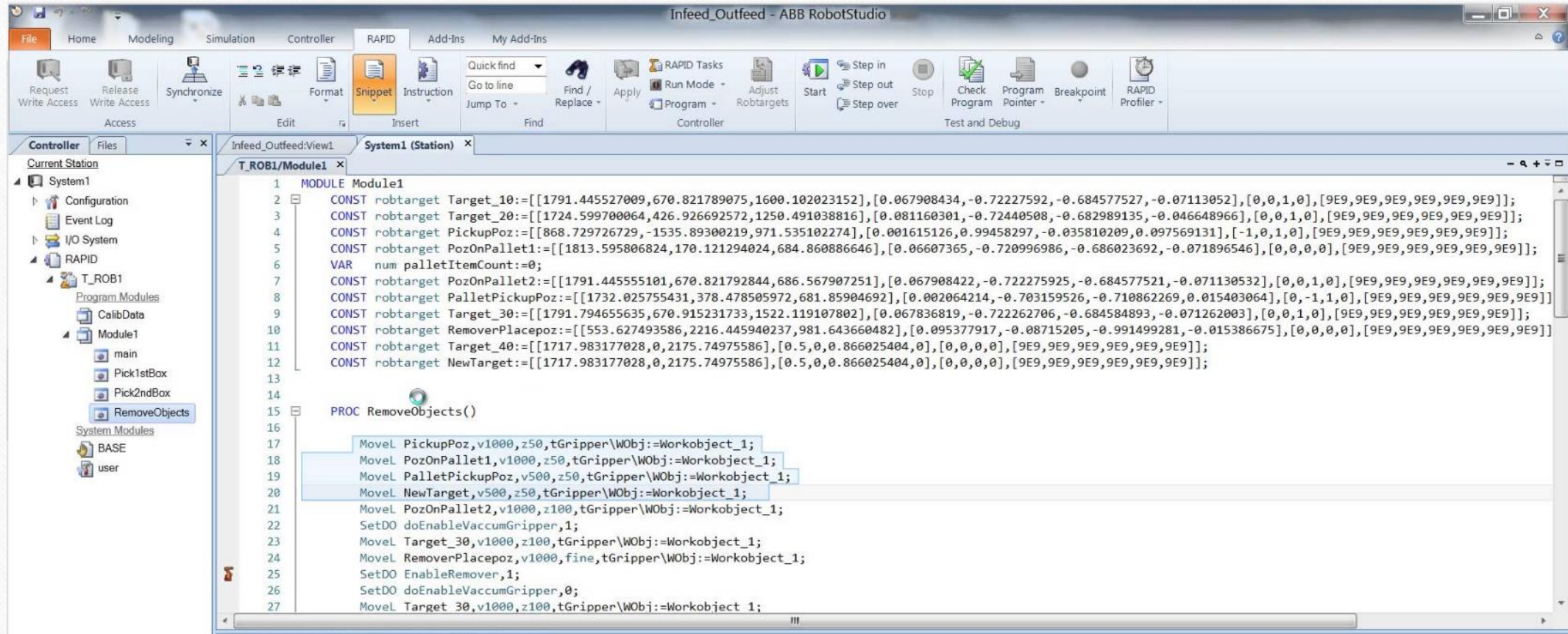
Suction Cup

# Software

---

- Robot Studio
  - Available on Campus computers & for a 30-day free trial ([Trial Download](#))
  - Simulation and offline programming software
- Rapid Code
  - High level language
- Other coding experience is not a requirement, but may be useful
  - Examples: Python, LabView, OpenCV, Matlab ...

# Robot Studio/Rapid Example



The screenshot displays the ABB RobotStudio interface. The main window shows a RAPID program for a robot module named 'T\_ROB1/Module1'. The program consists of several constant declarations for target positions and a procedure named 'RemoveObjects()'. The 'RemoveObjects()' procedure contains a sequence of 'MoveL' commands to move the gripper to various workobject positions and 'SetDO' commands to control the vacuum gripper.

```
1 MODULE Module1
2   CONST robtarget Target_10:=[[1791.445527009,670.821789075,1600.102023152],[0.067908434,-0.72227592,-0.684577527,-0.07113052],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
3   CONST robtarget Target_20:=[[1724.599700064,426.926692572,1250.491038816],[0.081160301,-0.72440508,-0.682989135,-0.046648966],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
4   CONST robtarget PickupPoz:=[[868.729726729,-1535.89300219,971.535102274],[0.001615126,0.99458297,-0.035810209,0.097569131],[-1,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
5   CONST robtarget PozOnPallet1:=[[1813.595806824,170.121294024,684.860886646],[0.06607365,-0.720996986,-0.686023692,-0.071896546],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
6   VAR num palletItemCount:=0;
7   CONST robtarget PozOnPallet2:=[[1791.445555101,670.821792844,686.567907251],[0.067908422,-0.722275925,-0.684577521,-0.071130532],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
8   CONST robtarget PalletPickupPoz:=[[1732.025755431,378.478505972,681.85904692],[0.002064214,-0.703159526,-0.710862269,0.015403064],[0,-1,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
9   CONST robtarget Target_30:=[[1791.794655635,670.915231733,1522.119107802],[0.067836819,-0.722262706,-0.684584893,-0.071262003],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
10  CONST robtarget RemoverPlacepoz:=[[553.627493586,2216.445940237,981.643660482],[0.095377917,-0.08715205,-0.991499281,-0.015386675],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
11  CONST robtarget Target_40:=[[1717.983177028,0,2175.74975586],[0.5,0,0.866025404,0],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
12  CONST robtarget NewTarget:=[[1717.983177028,0,2175.74975586],[0.5,0,0.866025404,0],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
13
14
15  PROC RemoveObjects()
16
17    MoveL PickupPoz,v1000,z50,tGripper\WObj:=Workobject_1;
18    MoveL PozOnPallet1,v1000,z50,tGripper\WObj:=Workobject_1;
19    MoveL PalletPickupPoz,v500,z50,tGripper\WObj:=Workobject_1;
20    MoveL NewTarget,v500,z50,tGripper\WObj:=Workobject_1;
21    MoveL PozOnPallet2,v1000,z100,tGripper\WObj:=Workobject_1;
22    SetDO doEnableVacuumGripper,1;
23    MoveL Target_30,v1000,z100,tGripper\WObj:=Workobject_1;
24    MoveL RemoverPlacepoz,v1000,fine,tGripper\WObj:=Workobject_1;
25    SetDO EnableRemover,1;
26    SetDO doEnableVacuumGripper,0;
27    MoveL Target_30,v1000,z100,tGripper\WObj:=Workobject_1;
```

# TA's and Scheduling

---

- Ryan Mocadlo – Make appointment by email (mocad@wpi.edu)
- Adam Gatehouse – Make appointment by email (ajgatehouse@wpi.edu)
- Project/Lab Groups:
  - Please create your groups on Canvas by **today at 5pm**
  - Sign up for 2, 1 hour lab blocks using the google sheets link on Canvas
    - [Schedule](#)
- TA's will be available to answer any questions during lab hours



# Report

---

- To be submitted on Canvas only (no hard copies) a week after your second session
- Rubrics for each lab posted week before lab
- Grade given will be either A, B, or C
  - Allowed one resubmission per lab
  - Grade will be increased by one letter if the group is able to address the graders comments
- Two strong example reports are discussed below (more examples uploaded to canvas)

# Ordering of Parts

---

- No budget for purchasing of parts
- Variety of parts already in lab
- Work with TA's to find best supplier
- Ability to use machine shop for custom parts
  - TA's can provide help on the machines in the shop
  - Link for necessary CAD files: [CAD Files](#)
- Do not breakdown, or remove items from the lab that are not your own

# Safety

---

- Must take basic safety quiz before entering the machine shop
  - Find on [mfelabs.org](http://mfelabs.org)
  - Must gain Advanced User status to use the additional shop machinery
    - Ask TA for more information
- Will gain Project Space User access
  - Allows for access to the ABB robot whenever a Lab Monitor is present

# Past Project (Industrial)

---

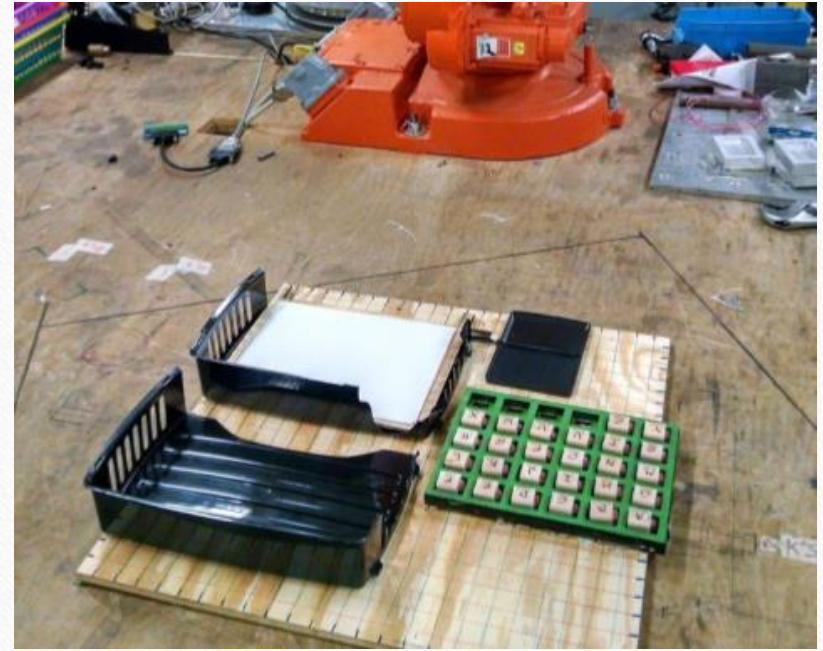
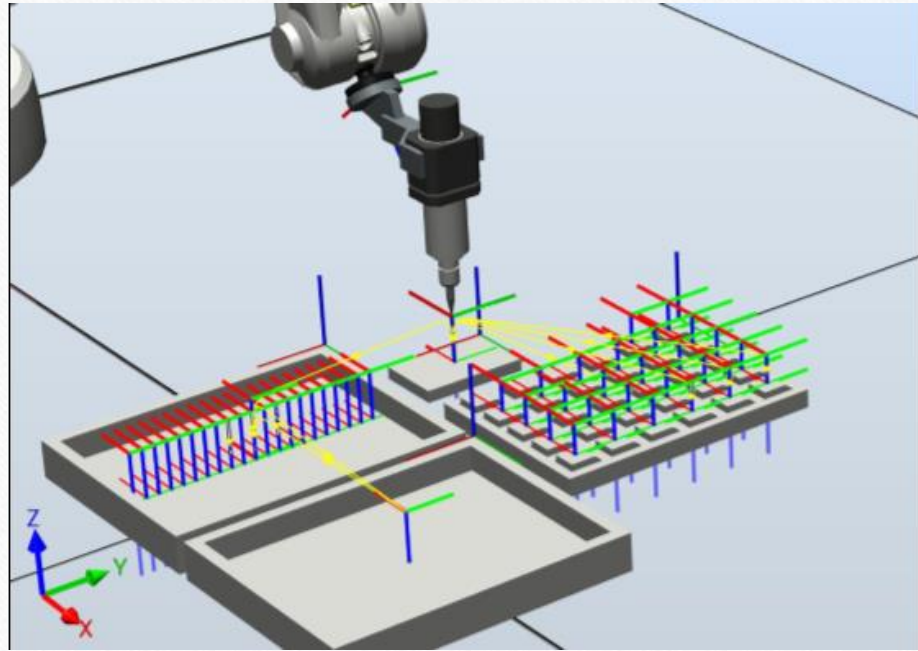
For this project, we decided to have the ABB IRB 1600 Industrial Robot stamp different words onto a piece of paper depending on user input. Our inspiration for this arose from the Sorority Recruitment season, which requires decorative name tags for all of the sisters for each round. This means that someone must write down all 100+ names for each round, for three rounds. We decided to use this project to explore an opportunity to automate this process by simply inputting names and having the robot stamp the input. The challenge in this task is that the program and robot would have to complete the following:

- Split the string input into individual characters
- Locate the stamp with the correct first character in the workspace
- Lift the stamp and stamp it onto an ink pad
- Lift the stamp again and stamp it onto the piece of paper in the correct location
- Place the stamp back in the correct holding location
- Repeat this process with the next character in the string
- At the end of the process, pick up the piece of paper and place it into the “finished” bin



# Past Project (Industrial)

---



# Past Project (Robotics)

---

For this project, an ABB IRB1600 industrial robot was configured to play a game of tic tac toe with a human player. The game pieces were 5 pink and 5 green cubes. An Arduino Uno connected to a Pixy camera was able to determine which cells of the 3x3 playing grid were occupied by which color cube. To simplify the problem, the robot was always the first player and always used the green cubes. The Arduino script chose the optimal grid cell to place the cube and output it as a number 0-8 corresponding to each grid cell. A LabVIEW program interpreted the Arduino output into a format usable by the NI USB-6525 DAQ. The RAPID code then determined which path to follow based on the status of the input lines of the NI DAQ. The game ended when a player got three game pieces in a row or all nine grid cells are occupied. The objective of this project is to program the robot to autonomously play tic tac toe with a human player using color detection. This project was successfully completed with the robot playing against a human player. Using predictive tic tac toe logic, the robot never lost a game against a human. The Pixy, Arduino, LabVIEW, and NI DAQ components were successfully able to communicate to complete the task.

# Past Project (Robotics)

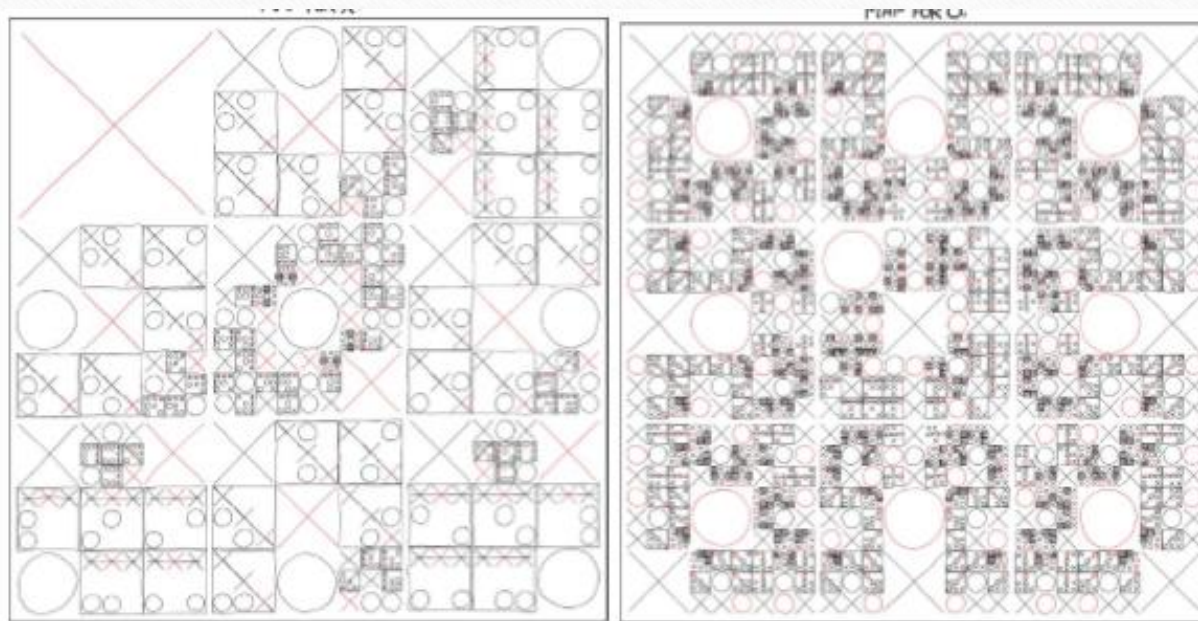
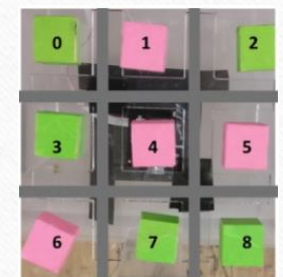
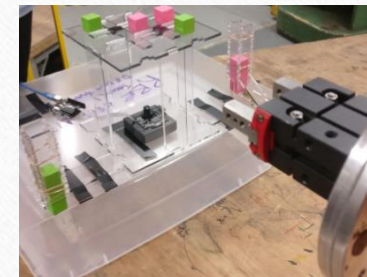
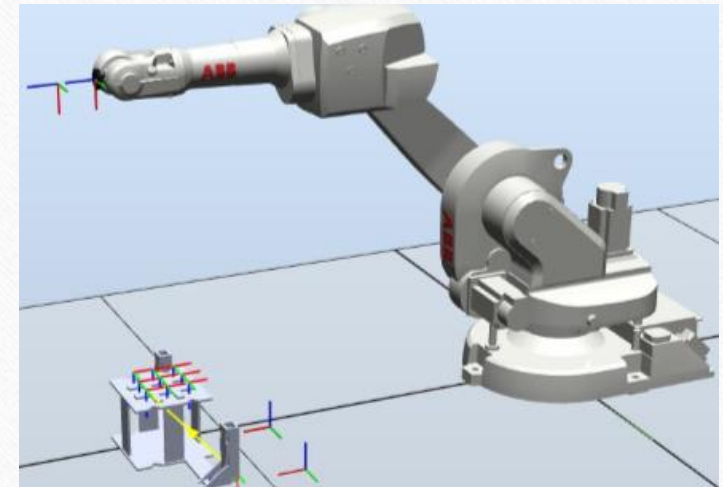


Figure 11: XKCD map for determining moves that result in win or tie



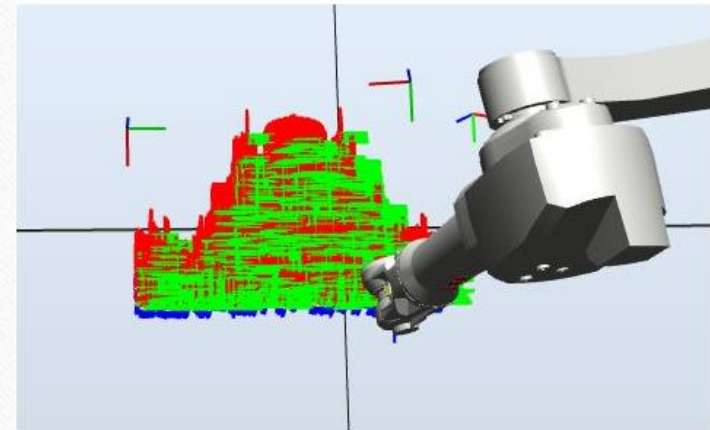
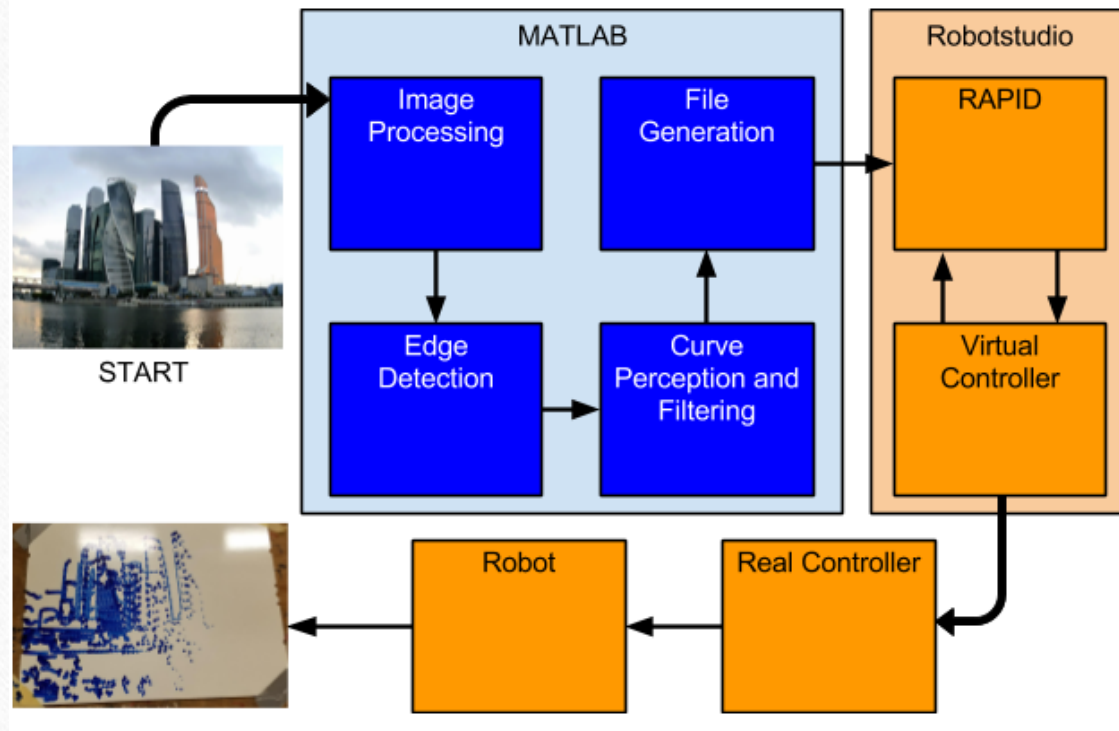
# Past Project (Robotics)

---

Using robots to draw is an emerging artform. To expand this field, a method for sketching any image, on a dry erase board, using an ABB IRB 1600 industrial robot, was developed. The four tasks to complete this goal include: developing the mechanical fixturing for the marker, writing path perception algorithms, porting the computer vision data to the robot's proprietary controller, and a full system integration and test. The provided 3-pronged radial gripper with custom 3D printed finger was used to grasp an Expo brand, bullet-tipped, dry erase marker. An 18 x 24 inch whiteboard was featured in the workspace to draw upon. To perceive to curve to sketch form the image, two main procedures were used from MATLAB's image processing toolbox. First, canny edge transform was performed on the reduced grayscale image. Then a breadth-first search algorithm was developed to generate a collection of vectors representing a list of paths, with each path being made up of groupings of coordinates. The next step was to import the paths into the ABB proprietary software, RobotStudio, and RAPID programming language. To accomplish this, the path data was written to a text file using RAPID syntax and pasted into RobotStudio. The final system was largely successful in sketching the image. Mechanically, there were issues with line thickness consistency as the board table is not entirely level as the marker was very rigidly held in the end of arm tool. Future iterations may include improved marker fixturing, further algorithm refinement, and multiple colors.



# Past Project (Robotics)



# Video Example Simulation

---

