# ME-GY 6933: Advanced Mechatronics (Spring 2020)

## Final Project

## Motion Tracking Camera System

Haoran Wu, Haoran Zhou, Anderson Cone

# Motivation - Core Problem

- Professors teaching remotely struggle to show students what they’re writing
- Must adjust camera regularly to show appropriate blackboard

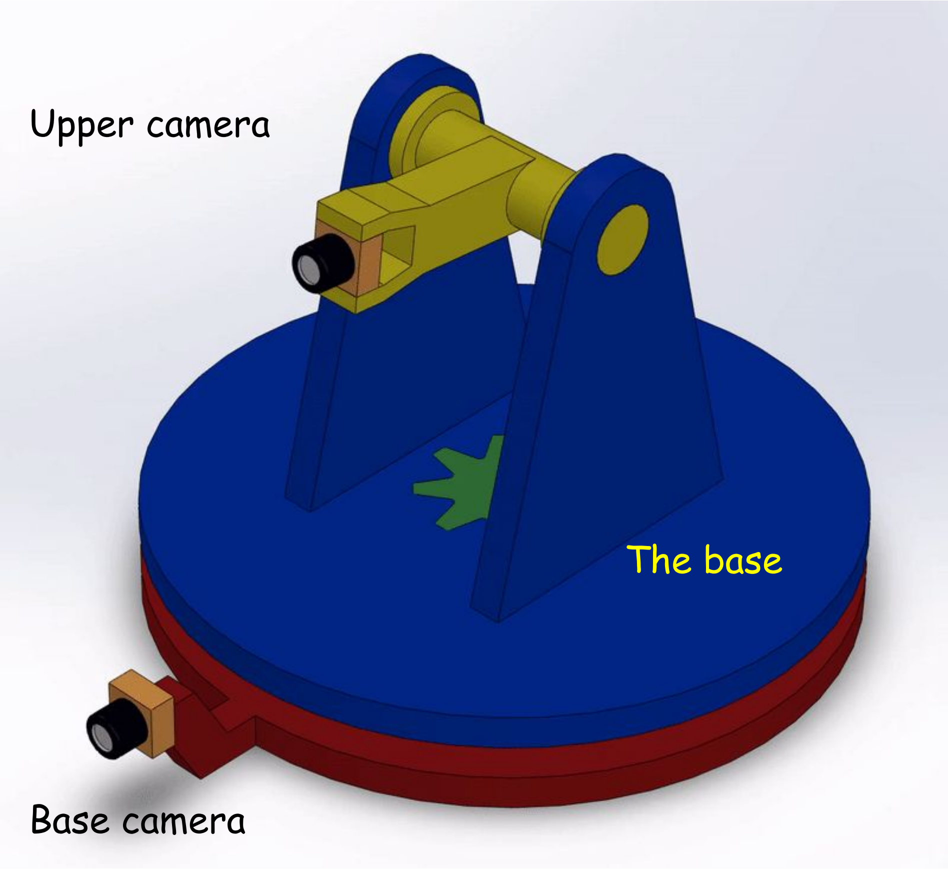
# Motivation - Additional Uses

## Entertainment

- Tracking camera, selfies
- Personal filming
- Game (Dodgeball)

## Security

- Motion sensitive, enhanced security monitoring ability (Home, Childminding, etc.)
- Switching to manually control, enable user to check by themselves

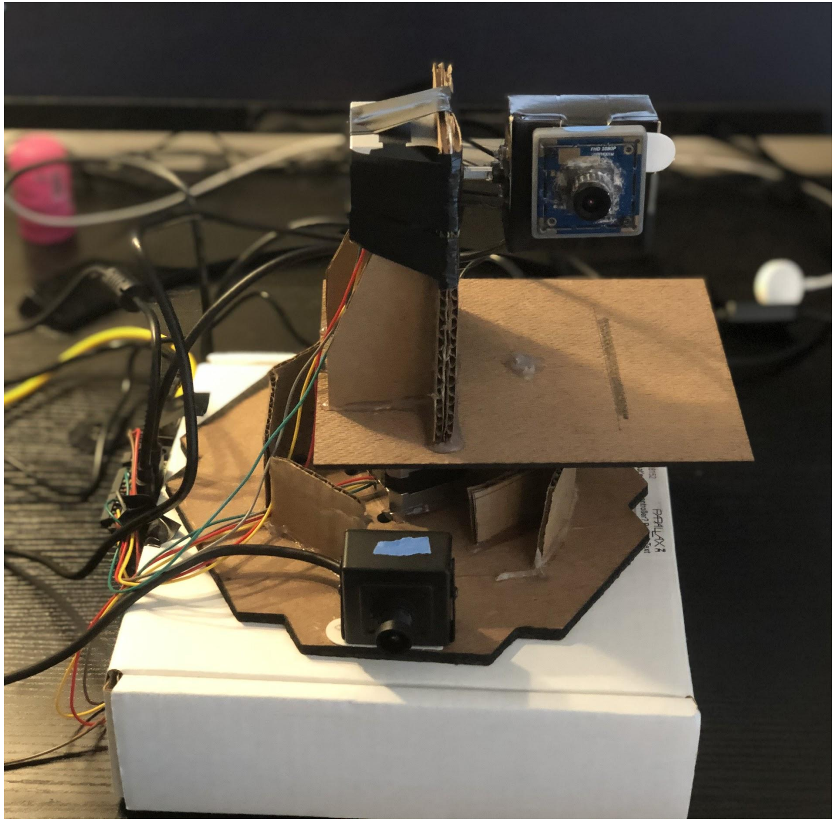


# Introduction - Motion Tracking Camera

**2 modes:**

(1) **Motion tracking mode**: Camera + OpenCV track the subject; Raspberry Pi controls stepper motors move the camera autonomously to focus on the subject.

(2) **Interactive mode**: Arduino and hand-held buttons send signals to Raspberry Pi to control the stepper motors to move camera manually.



# Component List

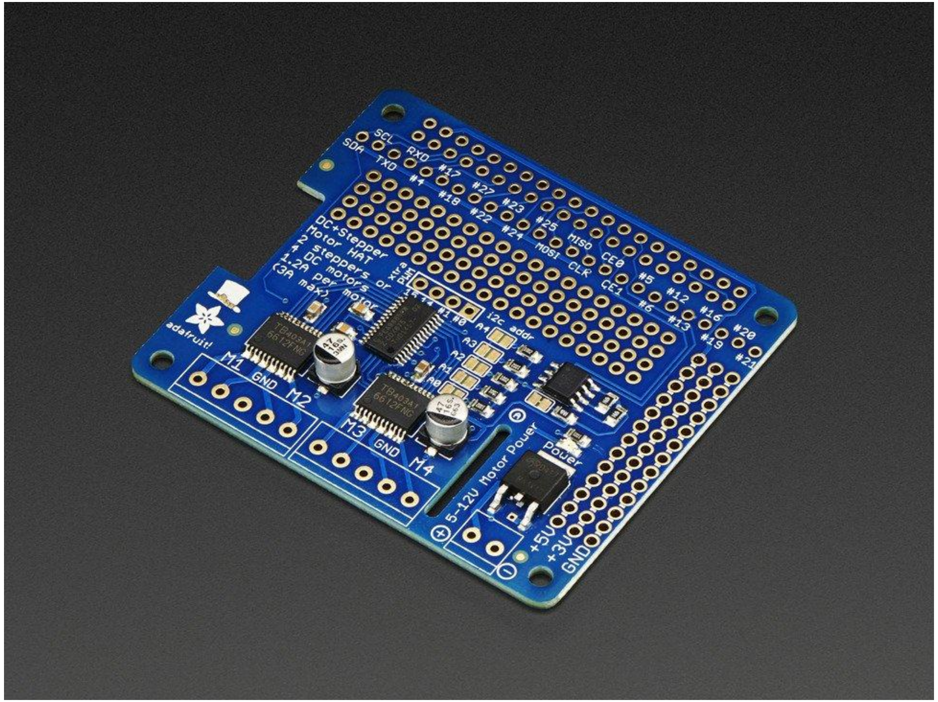
| Item # | Name | Quantity | Price (USD) |
|---|---|---|---|
| 1 | Cana Raspberry Pi 4 4GB Starter Kit | 1 | 99.99 |
| 2 | Arduino UNO board | 1 | 23.00 |
| 3 | Adafruit Stepper Motor | 2 | 28.00 |
| 4 | Adafruit Motor Hat for Raspberry Pi | 1 | 22.50 |
| 5 | USB Webcam 1080p | 2 | 50.00 |
| 6 | Voltage converter | 1 | 6.50 |
| 7 | Buttons | 4 | 1.00 |
| 8 | Wood board | 2 | 4.00 |
| 9 | Jump wires | / | / |
| 10 | Portable charger | 1 | 18.77 |
| 11 | Standoffs for Pi HATS | 1 | 0.75 |
| | | **Total** | **242.53** |

# Main Component - Stepper Motor HAT

- Since the Raspberry Pi does not have a lot of PWM pins, we use a fully-dedicated PWM driver chip on board to both control motor direction and speed.

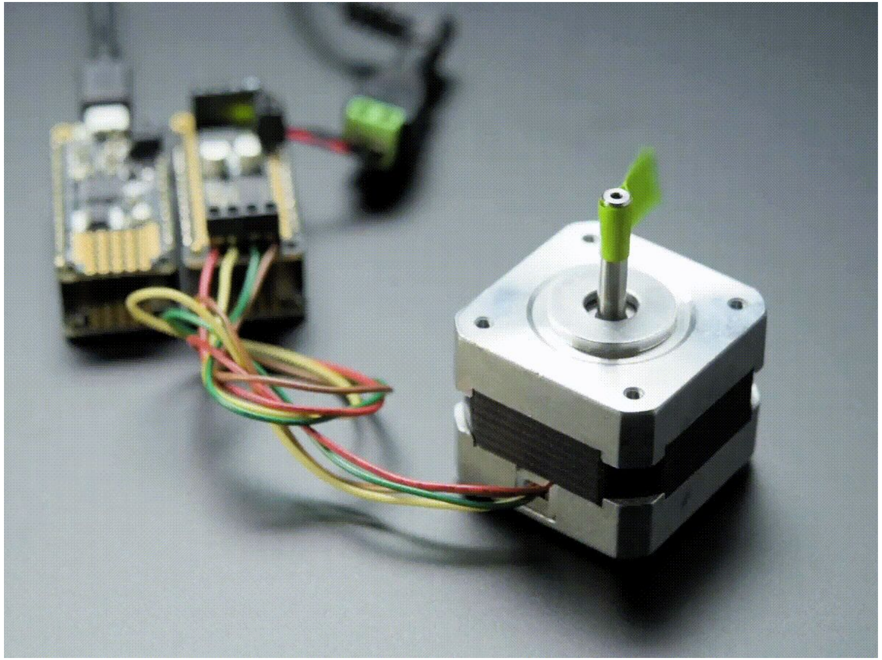
- This chip handles all the motor and speed controls over I2C. Only two pins (SDA & SCL) are required to drive the multiple motors, and since it's I2C you can also connect any other I2C devices or HAT to the same pins.

- External 12V to power the HAT

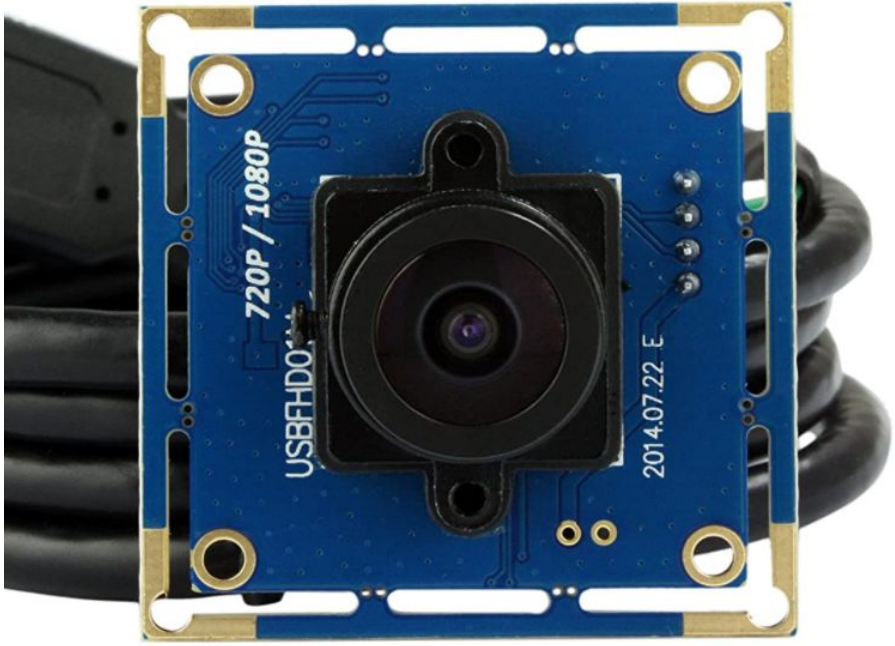


Adafruit DC & Stepper Motor HAT for Raspberry Pi

# Main Component - Stepper Motor and Camera

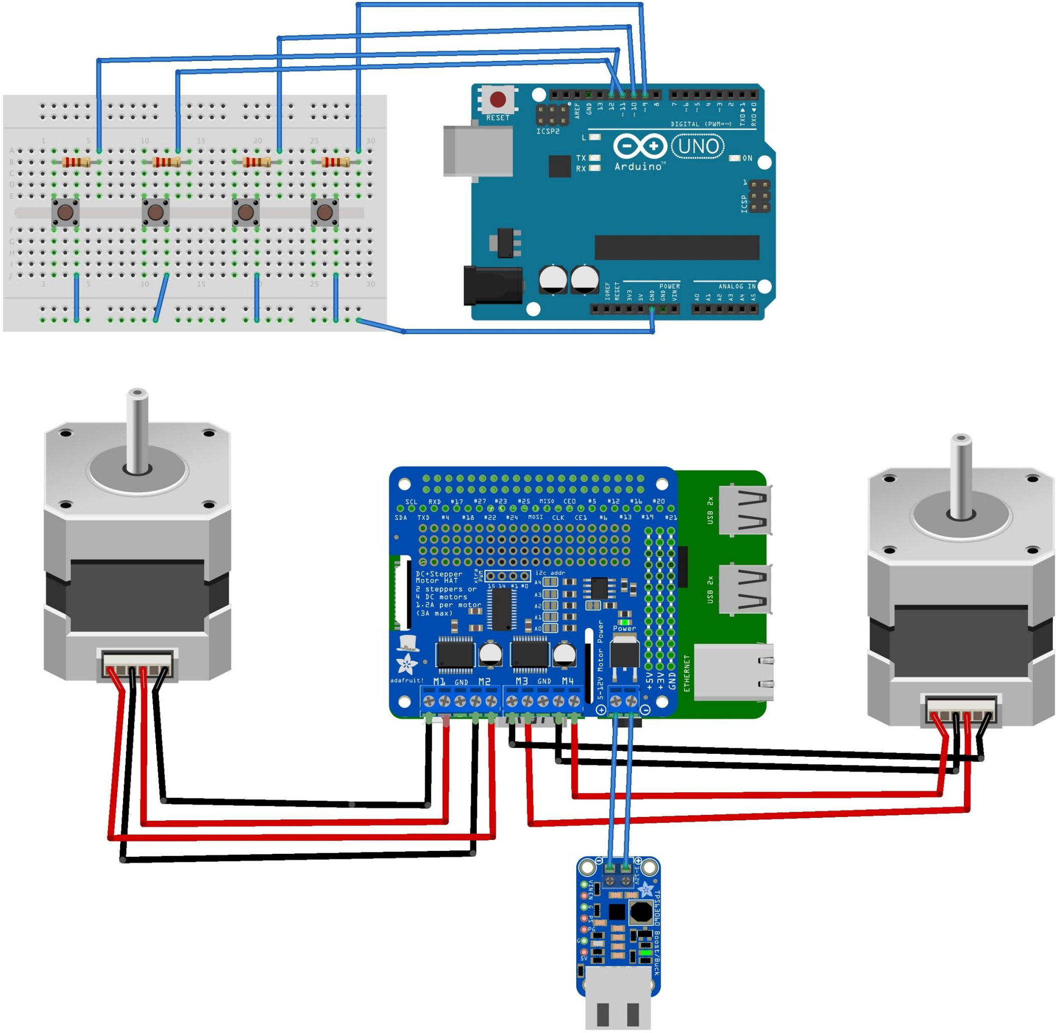


Stepper motor - NEMA-17 size - 200 steps/rev, 12V 350mA



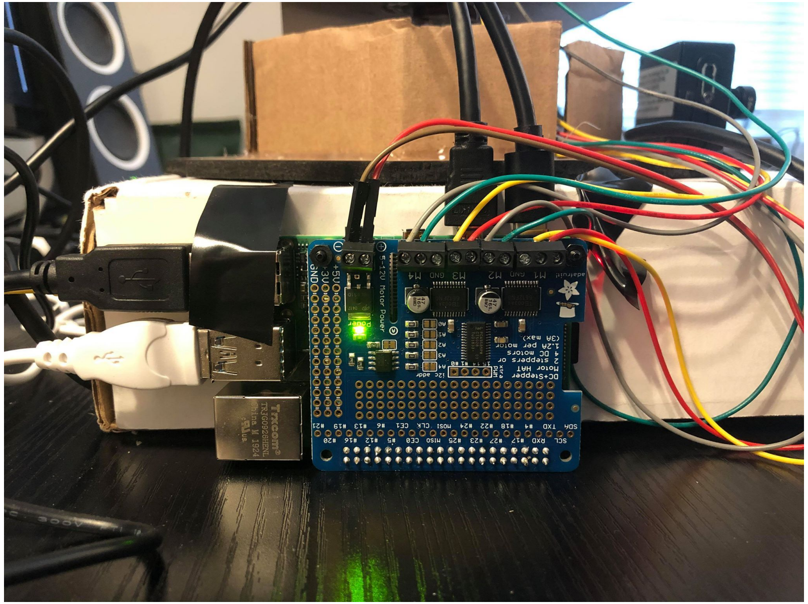
ELP USB with Camera 2.1mm Lens

# Circuit Diagram

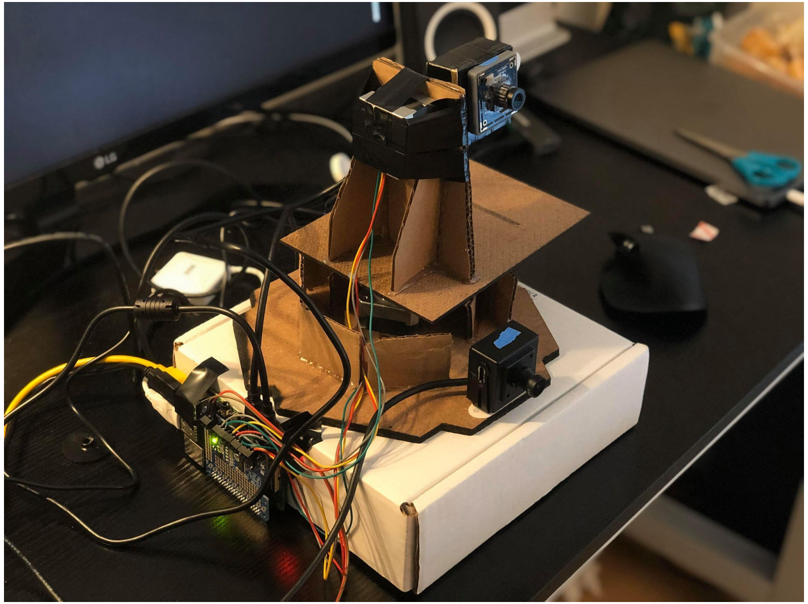


Arduino and Pi are using serial communication via USB cable

# Key Features: Hardware Anatomy

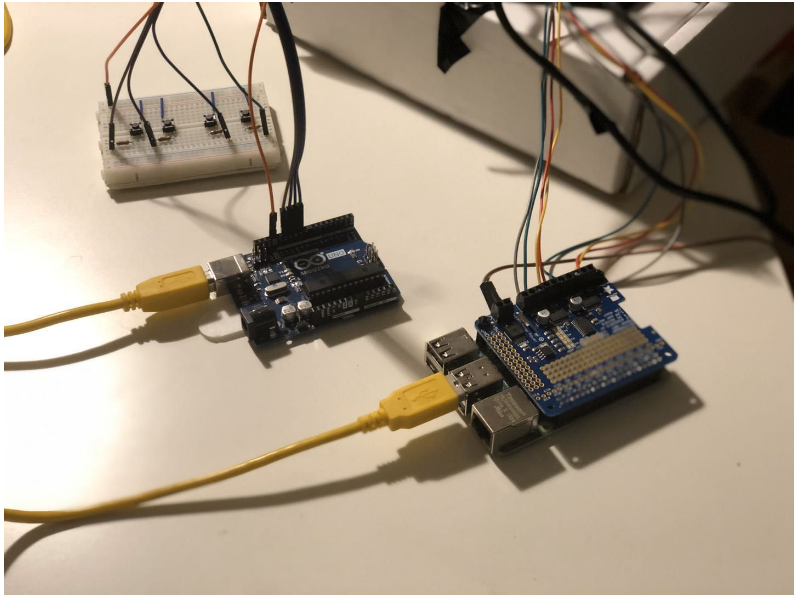


Raspberry Pi with Adafruit motor HAT

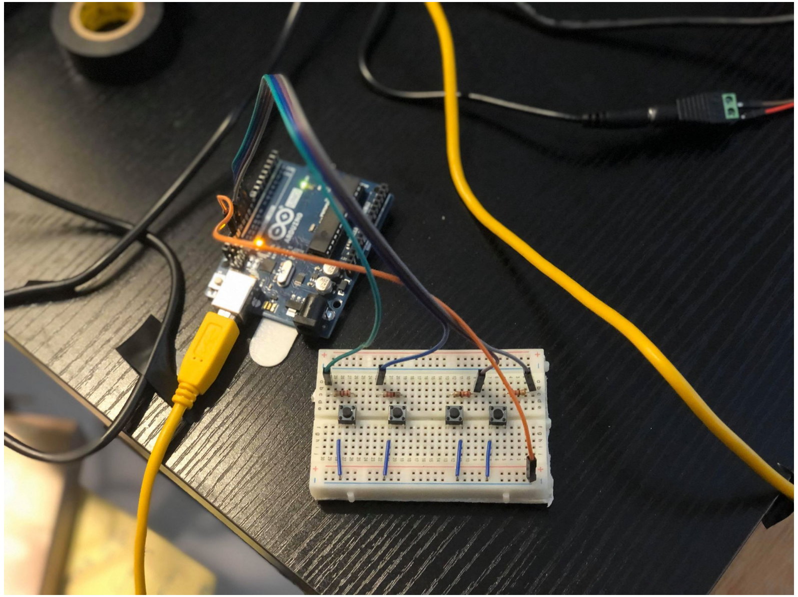


Base structure, web cameras, Adafruit stepper motors, Raspberry Pi and USB dock

# Key Features: Hardware Anatomy



Communicate between Arduino UNO and Raspberry Pi 4B



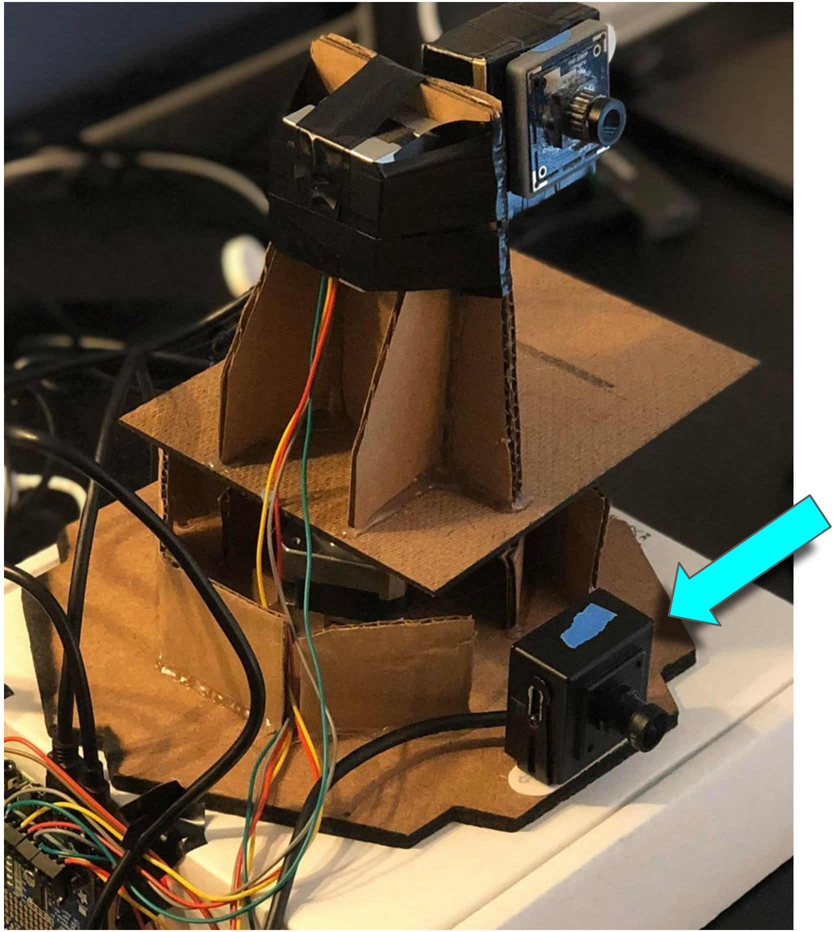
Left, right, up and down buttons for interactive mode control

# Test - Motion Detection

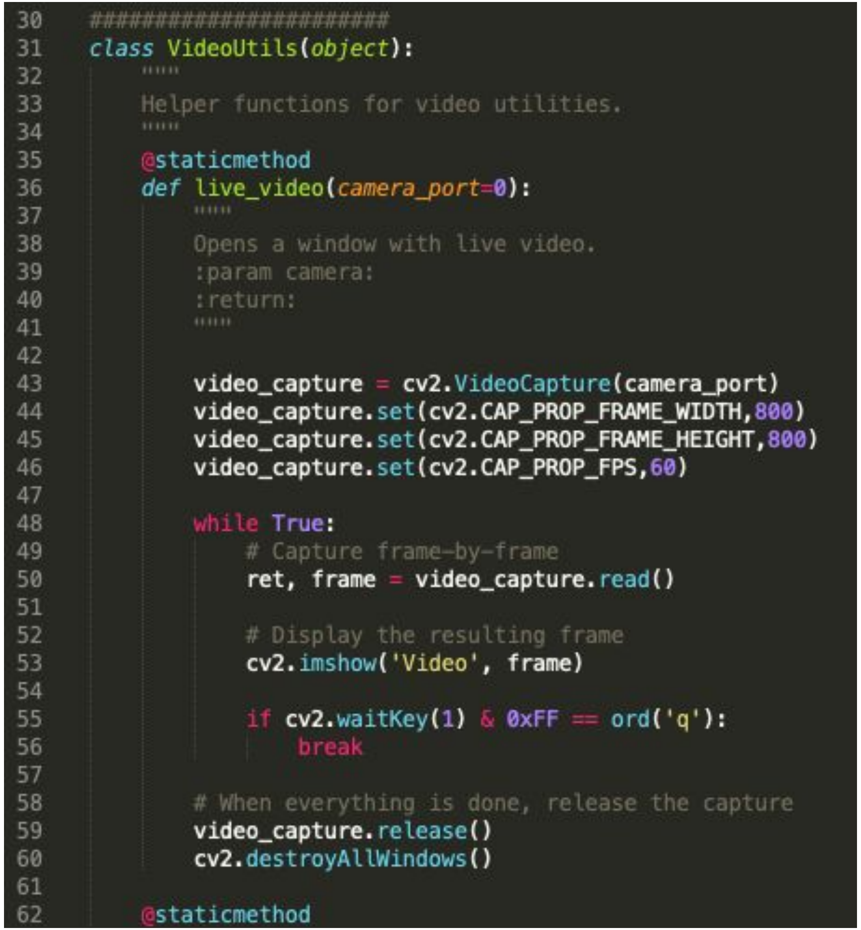
- **Find contours** by applying greyscale, blur, threshold, and dilate

- **Compare** each frame of the captured video flow with the initial frame.

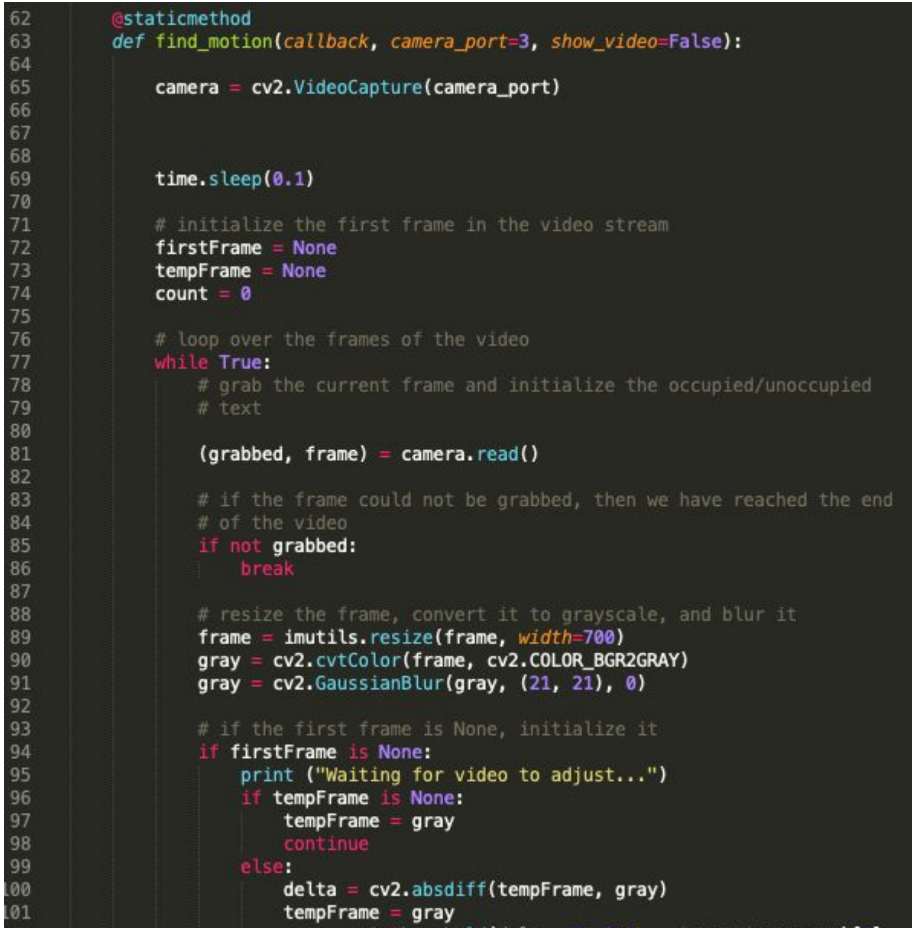
- All frames are captured from the base camera.



# Test - Motion Detection

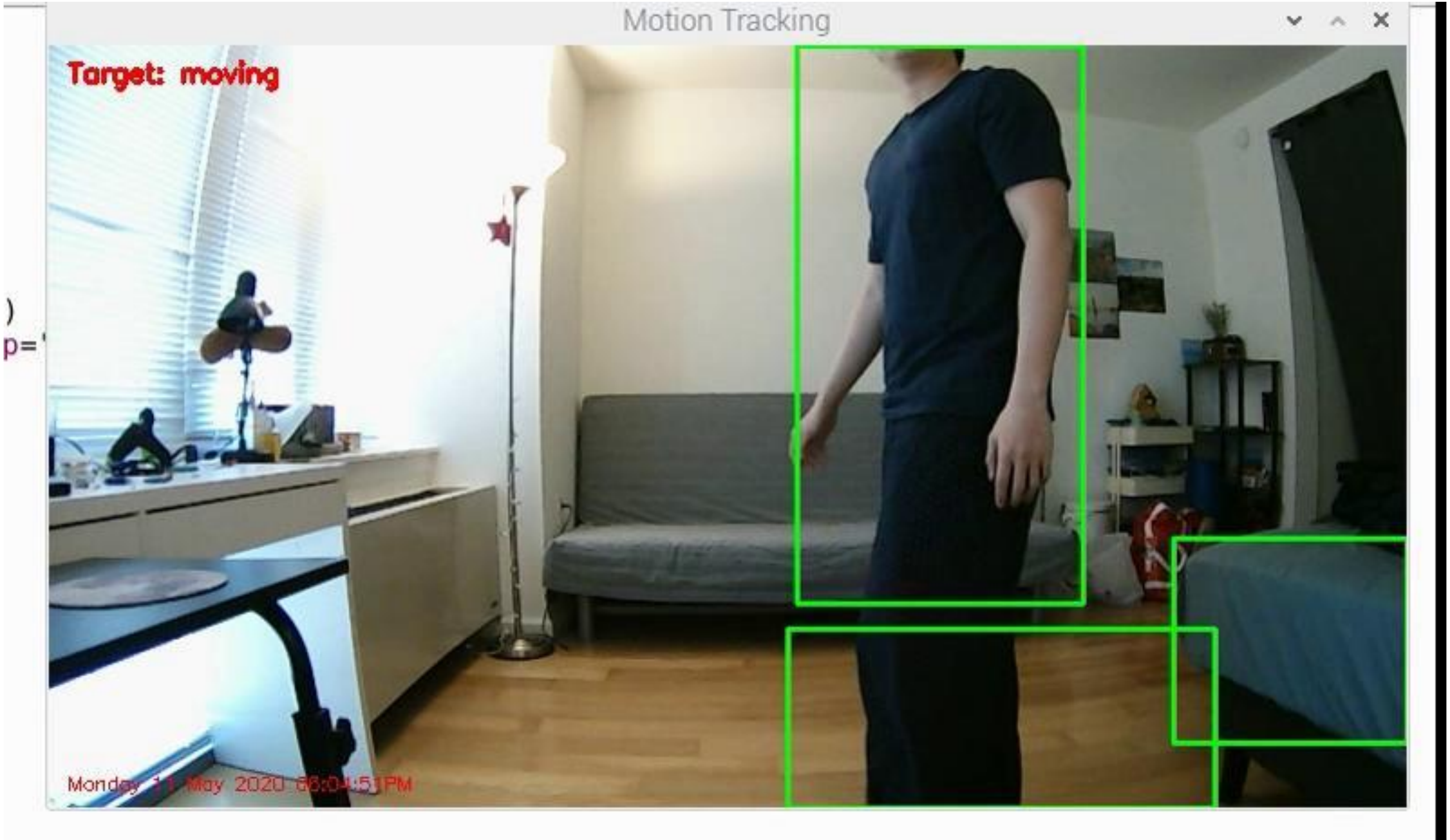


```python
########################
class VideoUtils(object):
    """
    Helper functions for video utilities.
    """
    @staticmethod
    def live_video(camera_port=0):
        """
        Opens a window with live video.
        :param camera:
        :return:
        """

        video_capture = cv2.VideoCapture(camera_port)
        video_capture.set(cv2.CAP_PROP_FRAME_WIDTH,800)
        video_capture.set(cv2.CAP_PROP_FRAME_HEIGHT,800)
        video_capture.set(cv2.CAP_PROP_FPS,60)

        while True:
            # Capture frame-by-frame
            ret, frame = video_capture.read()

            # Display the resulting frame
            cv2.imshow('Video', frame)

            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

        # When everything is done, release the capture
        video_capture.release()
        cv2.destroyAllWindows()

    @staticmethod
```



```python
    @staticmethod
    def find_motion(callback, camera_port=3, show_video=False):

        camera = cv2.VideoCapture(camera_port)



        time.sleep(0.1)

        # initialize the first frame in the video stream
        firstFrame = None
        tempFrame = None
        count = 0

        # loop over the frames of the video
        while True:
            # grab the current frame and initialize the occupied/unoccupied
            # text

            (grabbed, frame) = camera.read()

            # if the frame could not be grabbed, then we have reached the end
            # of the video
            if not grabbed:
                break

            # resize the frame, convert it to grayscale, and blur it
            frame = imutils.resize(frame, width=700)
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            gray = cv2.GaussianBlur(gray, (21, 21), 0)

            # if the first frame is None, initialize it
            if firstFrame is None:
                print ("Waiting for video to adjust...")
                if tempFrame is None:
                    tempFrame = gray
                    continue
                else:
                    delta = cv2.absdiff(tempFrame, gray)
                    tempFrame = gray
```

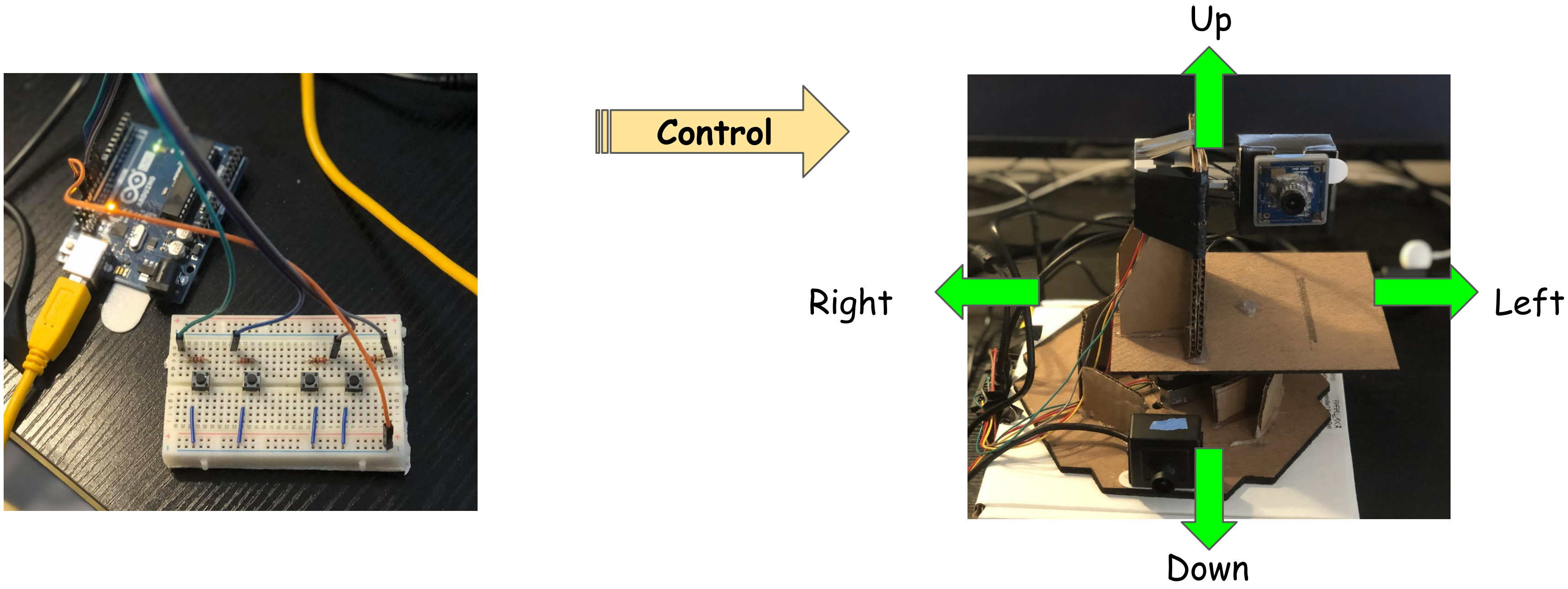
# Demo 1: Motion Detection



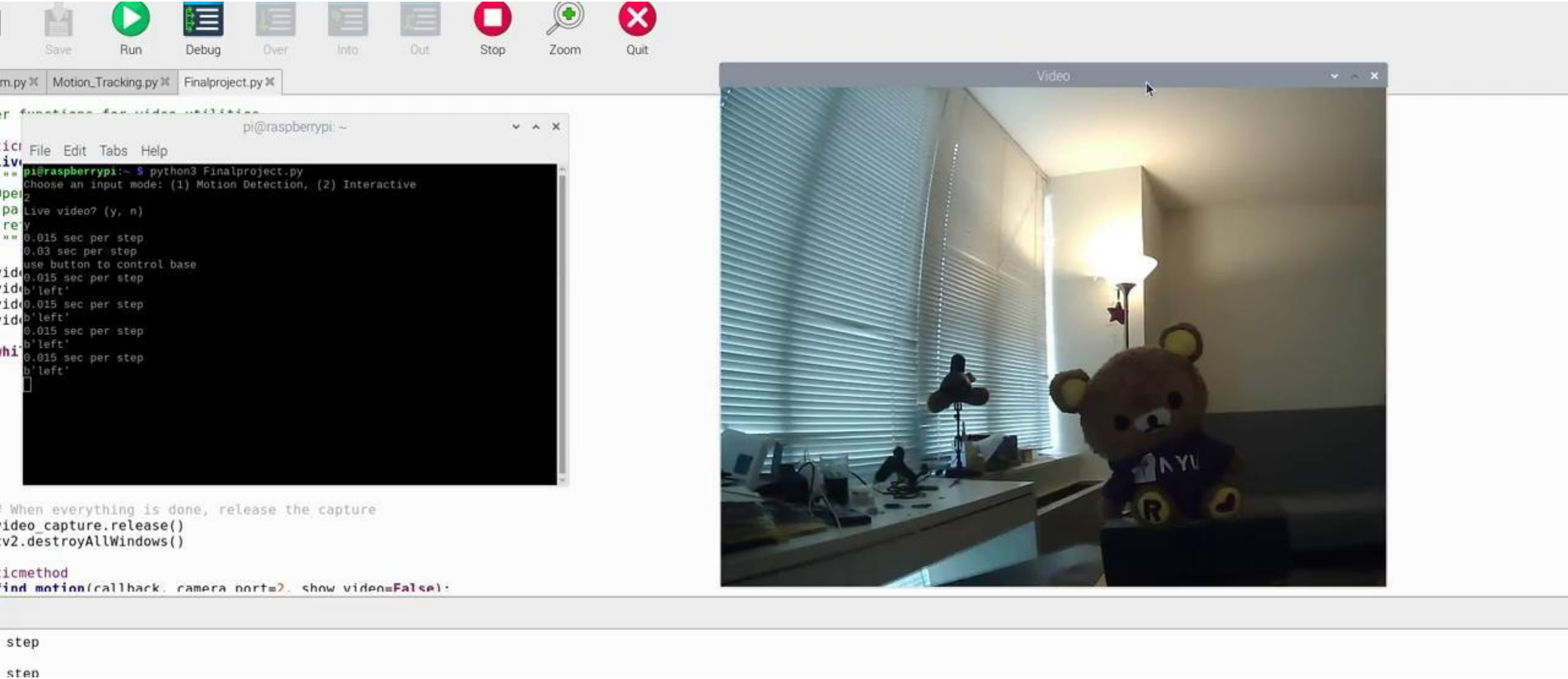
```
)
p='
```

# Prototype - Interactive Mode

**Interactive mode**: Arduino and buttons send signals to Raspberry Pi to control the stepper motors to move the upper camera manually



# Demo 2: Interactive Mode Controlled with Arduino



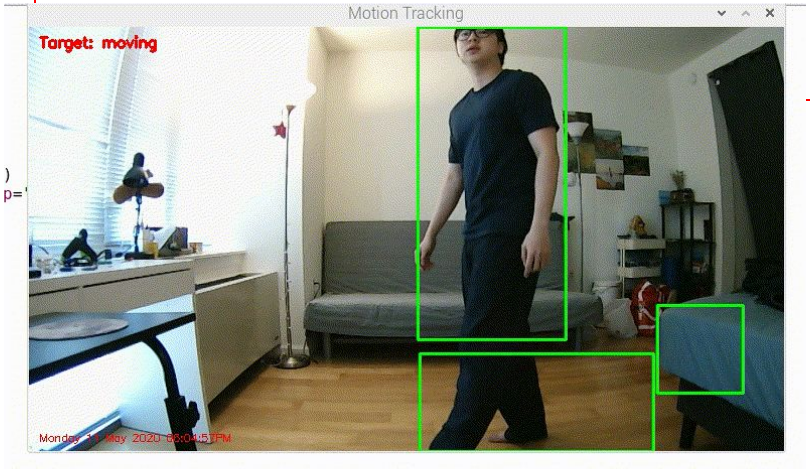
```
pi@raspberrypi:~ $ python3 Finalproject.py
Choose an input mode: (1) Motion Detection, (2) Interactive
2
Live video? (y, n)
y
0.015 sec per step
0.03 sec per step
use button to control base
0.015 sec per step
b'left'
0.015 sec per step
b'left'
0.015 sec per step
b'left'
0.015 sec per step
b'left'
```

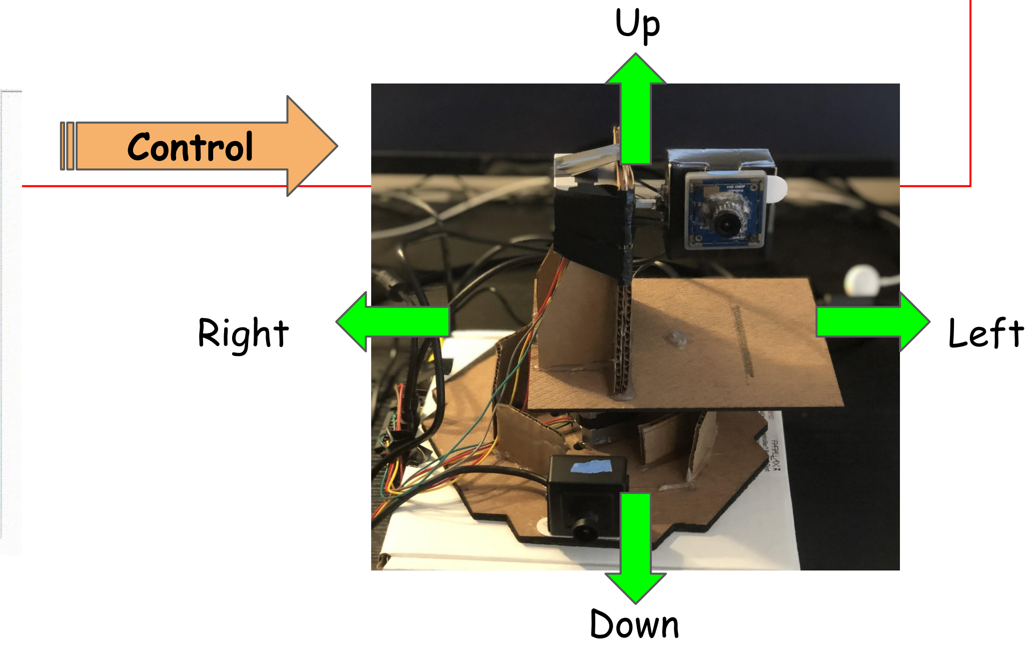
```
# When everything is done, release the capture
ideo_capture.release()
v2.destroyAllWindows()

icmethod
ind_motion(callback, camera_port=2, show_video=False):
```

```
step

step
```

# Prototype - Motion Tracking Mode

**Motion tracking mode**: using Base camera to track the motion, and the stepper motors are controlled to move the Upper camera autonomously to focus on the subject

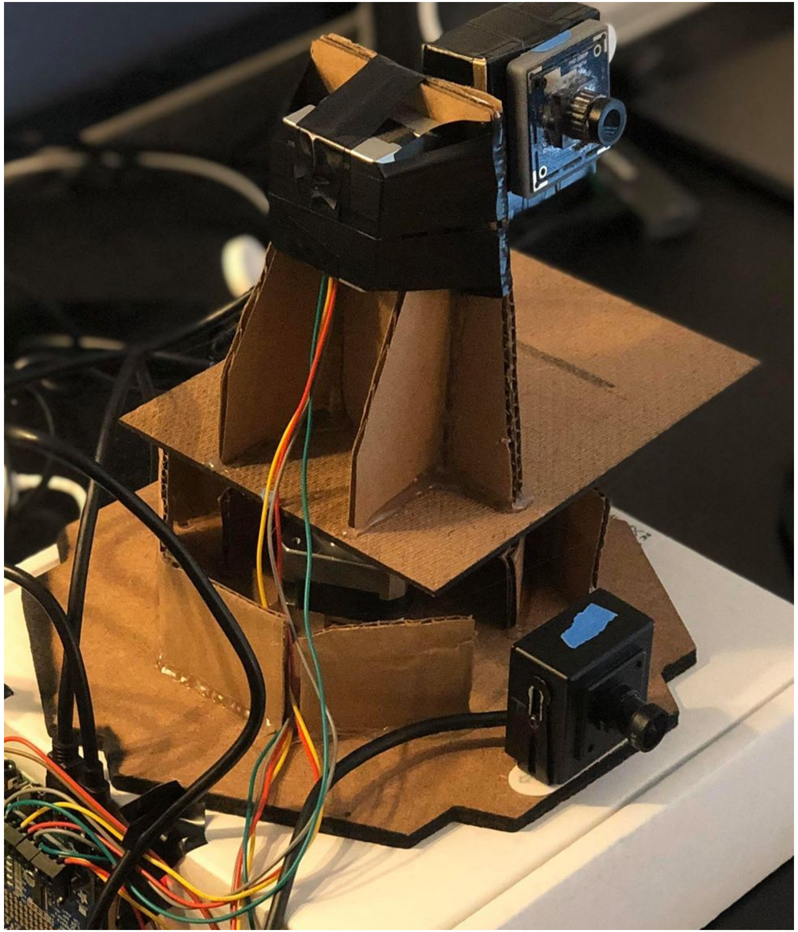




# Demo 3 - Motion Tracking

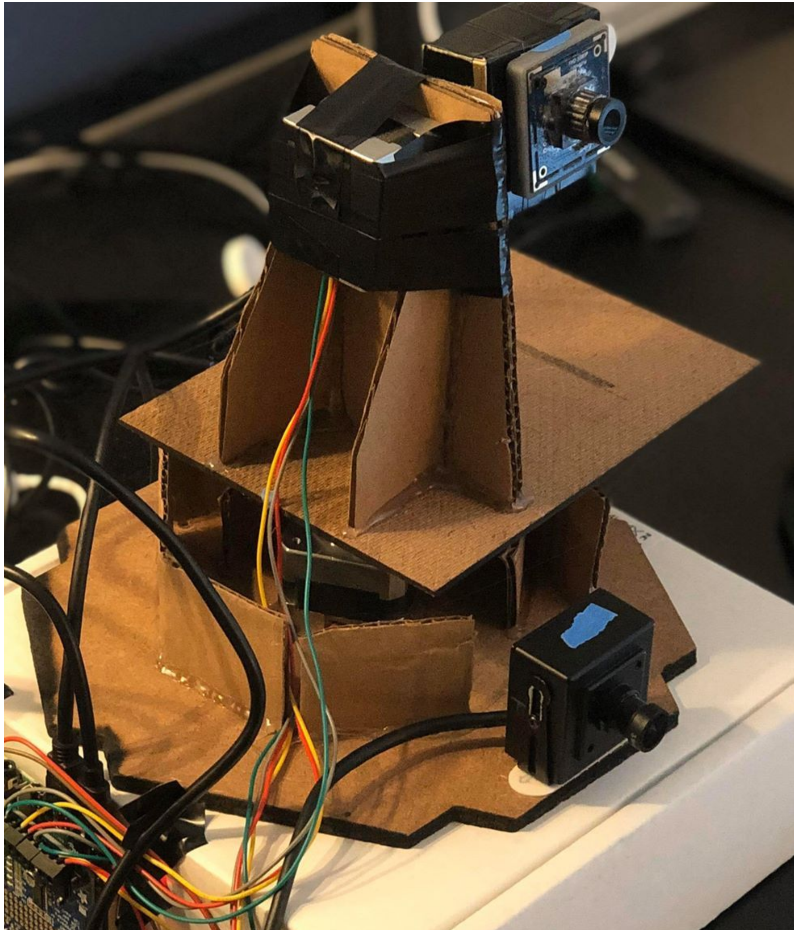
# Difficulties

- Control of Adafruit Stepper motor
- Lack of material to build structure
- Motion tracking algorithm affected by light conditions



# Future Work

- Replace wood/cardboard structure with 3D printed housing
- Replace Upper camera with smartphone
- Use RF modules for wireless connection between Arduino and Raspberry Pi
- Improve motion tracking algorithm to enhance reliability in low light conditions
- Add mode switch button to handheld user controls



# ME-GY 6933: Advanced Mechatronics (Spring 2020)

## Final Project

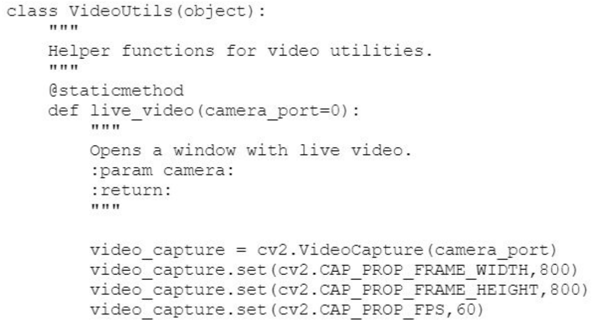
## Motion Tracking Camera System

Haoran Wu, Haoran Zhou, Anderson Cone

# Appendix

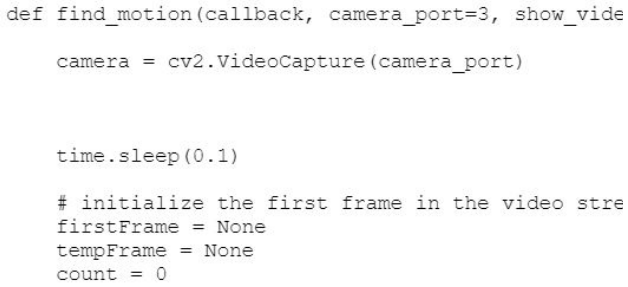
# Key Features:Code Anatomy

- Basic video display

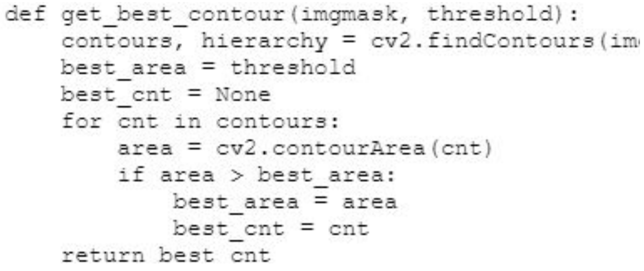


```
class VideoUtils(object):
    """
    Helper functions for video utilities.
    """
    @staticmethod
    def live_video(camera_port=0):
        """
        Opens a window with live video.
        :param camera:
        :return:
        """

        video_capture = cv2.VideoCapture(camera_port)
        video_capture.set(cv2.CAP_PROP_FRAME_WIDTH,800)
        video_capture.set(cv2.CAP_PROP_FRAME_HEIGHT,800)
        video_capture.set(cv2.CAP_PROP_FPS,60)
```

- Motion find



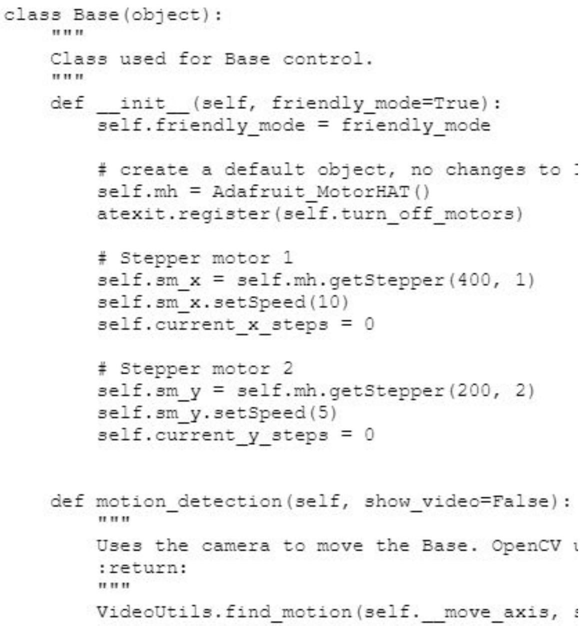
```
def find_motion(callback, camera_port=3, show_vide

    camera = cv2.VideoCapture(camera_port)



    time.sleep(0.1)

    # initialize the first frame in the video stre
    firstFrame = None
    tempFrame = None
    count = 0
```



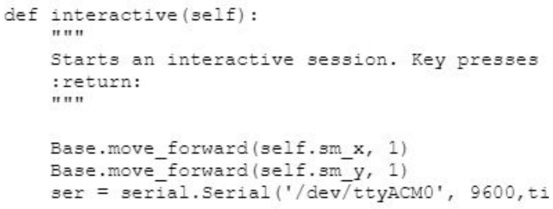
```
def get_best_contour(imgmask, threshold):
    contours, hierarchy = cv2.findContours(im
    best_area = threshold
    best_cnt = None
    for cnt in contours:
        area = cv2.contourArea(cnt)
        if area > best_area:
            best_area = area
            best_cnt = cnt
    return best_cnt
```

# Key Features:Code Anatomy

- Base control(Motion tracking mode and Interactive mode)

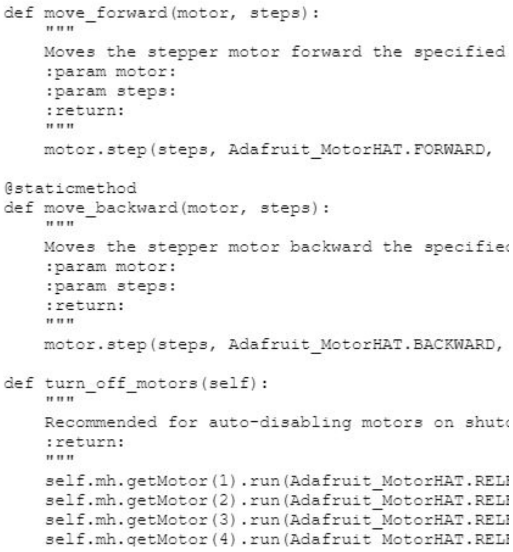


```
class Base(object):
    """
    Class used for Base control.
    """
    def __init__(self, friendly_mode=True):
        self.friendly_mode = friendly_mode

        # create a default object, no changes to
        self.mh = Adafruit_MotorHAT()
        atexit.register(self.turn_off_motors)

        # Stepper motor 1
        self.sm_x = self.mh.getStepper(400, 1)
        self.sm_x.setSpeed(10)
        self.current_x_steps = 0

        # Stepper motor 2
        self.sm_y = self.mh.getStepper(200, 2)
        self.sm_y.setSpeed(5)
        self.current_y_steps = 0


    def motion_detection(self, show_video=False):
        """
        Uses the camera to move the Base. OpenCV
        :return:
        """
        VideoUtils.find_motion(self.__move_axis,
```



```
def interactive(self):
    """
    Starts an interactive session. Key presses
    :return:
    """

    Base.move_forward(self.sm_x, 1)
    Base.move_forward(self.sm_y, 1)
    ser = serial.Serial('/dev/ttyACM0', 9600,ti
```

- Adafruit stepper motor control



```
def move_forward(motor, steps):
    """
    Moves the stepper motor forward the specified
    :param motor:
    :param steps:
    :return:
    """
    motor.step(steps, Adafruit_MotorHAT.FORWARD,

@staticmethod
def move_backward(motor, steps):
    """
    Moves the stepper motor backward the specifie
    :param motor:
    :param steps:
    :return:
    """
    motor.step(steps, Adafruit_MotorHAT.BACKWARD,

def turn_off_motors(self):
    """
    Recommended for auto-disabling motors on shut
    :return:
    """
    self.mh.getMotor(1).run(Adafruit_MotorHAT.REL
    self.mh.getMotor(2).run(Adafruit_MotorHAT.REL
    self.mh.getMotor(3).run(Adafruit_MotorHAT.REL
    self.mh.getMotor(4).run(Adafruit_MotorHAT.REL
```