THE FLYING, SPYING PI

AN AUTONOMOUS DRONE FOR AERIAL IMAGING AND IMAGE ANALYSIS

ABSTRACT

The Flying Pi Eye is an autonomous, flying drone with four rotors that lift a Raspberry Pi, GPS, and camera. The drone is intended for search and rescue situations. It creates aerial photography of an area, searches the photograph for any area of a particular color within a provided margin of error, and gives the GPS coordinates of the found areas. The drone is a controlled by a program written in Python that uses the Python Imaging Library to stitch together individual pictures to create the combined photo of the entire search area. The Python Imaging Library is also used to analyze the image to locate areas of the indicated color. Trigonometry is used to find the GPS coordinate of any pixel on the image given the GPS coordinate the picture was taken from and the altitude of the drone. This is used to calculate the GPS coordinates of sections of the image that match the color being searched for. To interact with the program a user connects the on board Raspberry Pi through ethernet to a network, sets the search area and color, and views the results through a web interface created with the Flask library in Python, clicks the start button, and disconnects the ethernet cable. Though currently navigated by the quadcopter, the Raspberry Pi, GPS, and camera could be attached to any flying vehicle in order to provide better range or save costs if a search and rescue department already had a drone and didn't need the quadcopter portion of this project.

INTRODUCTION AND TERMS

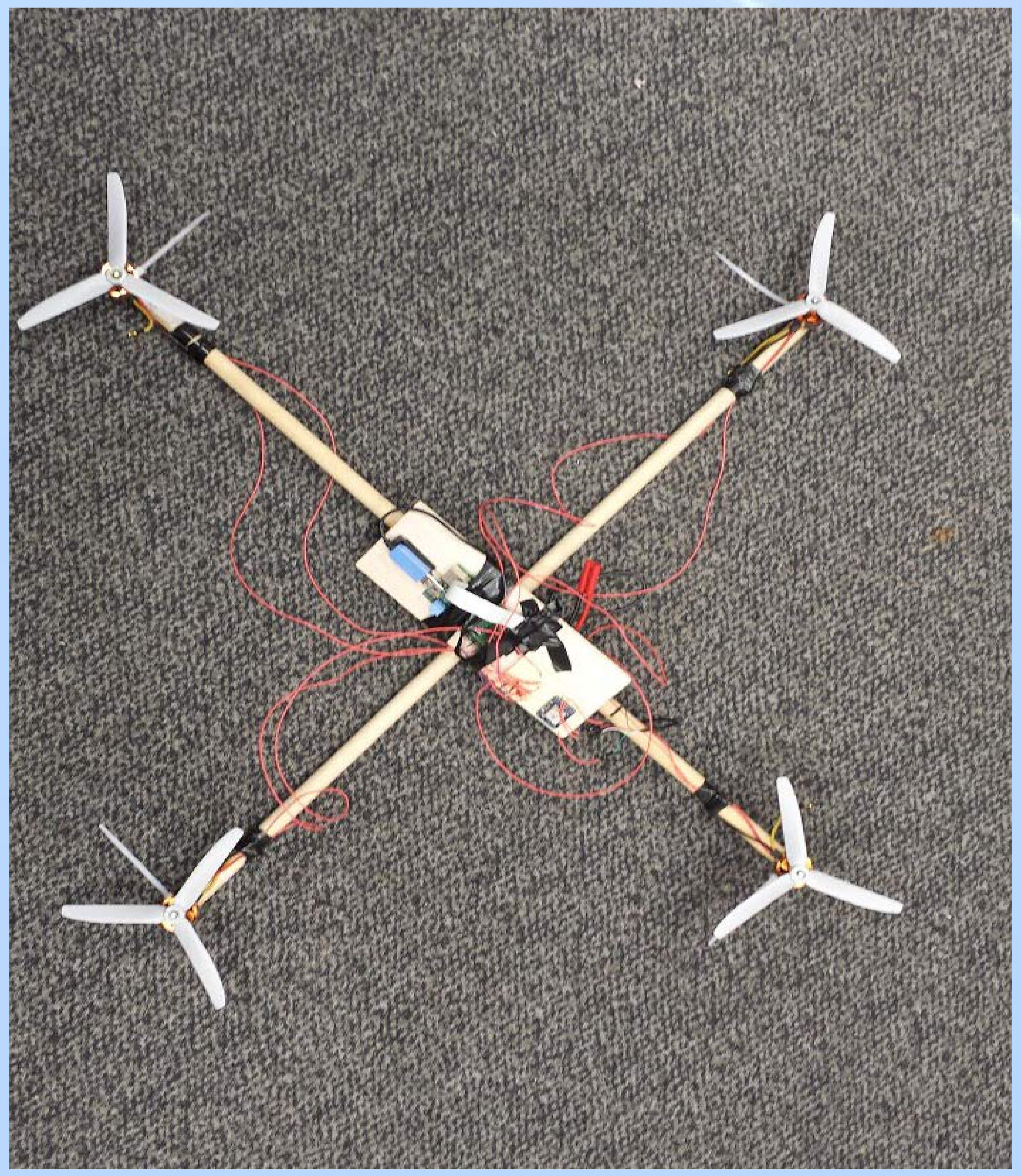
- A quadcopter is a helicopter propelled by four rotors, two spinning clockwise and two spinning counterclockwise.
- A Raspberry Pi is an inexpensive, low power computer, approximately the size of a credit card, with multiple General Purpose Input/Output pins used for interacting with other hardware.
- This project uses a Raspberry Pi to direct a quadcopter fitted with a camera and GPS to take aerial photography for image analysis.
- The images captured and analyzed by the Raspberry Pi are used to find the latitude and longitude of areas of interest identified through image analysis by interpolating the latitude and longitude from the edges of the picture.

PURPOSE

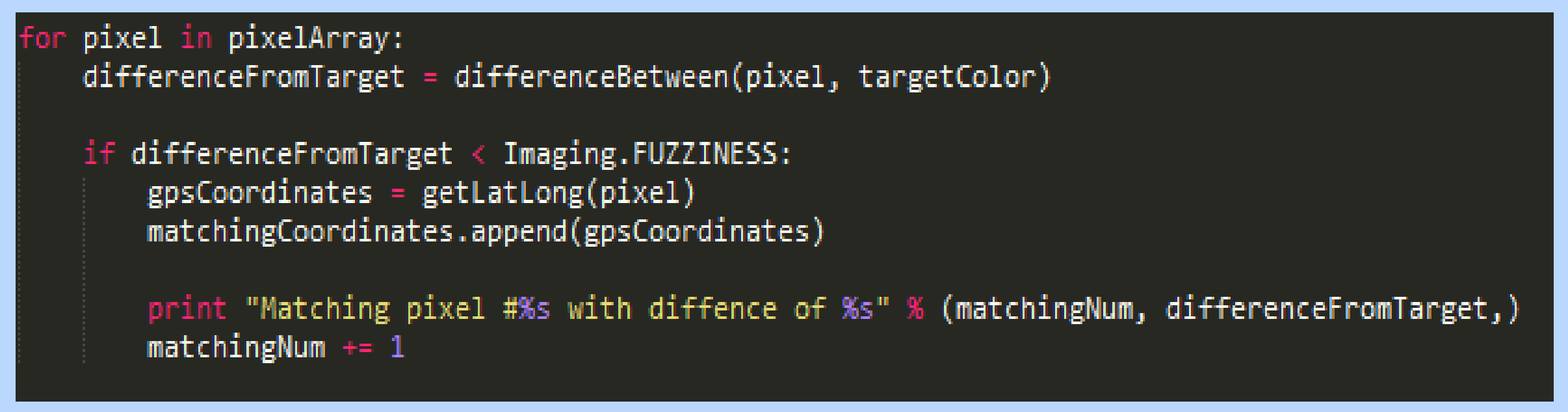
Current aerial drones are expensive. Search and rescue operations are not only expensive but dangerous for the rescuers. This project aims to build a device that functions as an inexpensive aid to rescuers searching for people and things, such as a lost hiker in rough terrain.

ENGINEERING GOALS

- Create a program to divide an area into an array of latitudes and longitudes at which to take pictures in order to capture the entire area, given that the pictures are taken at some designated height.
- Create a web based interface for this program using the Flask library.
- Run this program on a Raspberry Pi connected to a GPS and camera to automatically take pictures at the calculated coordinates.
- Create a quadcopter, directed by the Raspberry Pi, to carry the imaging module.
- Connect the quadcopter and Raspberry Pi to create a functioning unit capable of scanning an area and giving the latitude and longitude of places within the scanned area that match a given color.



Quadcopter

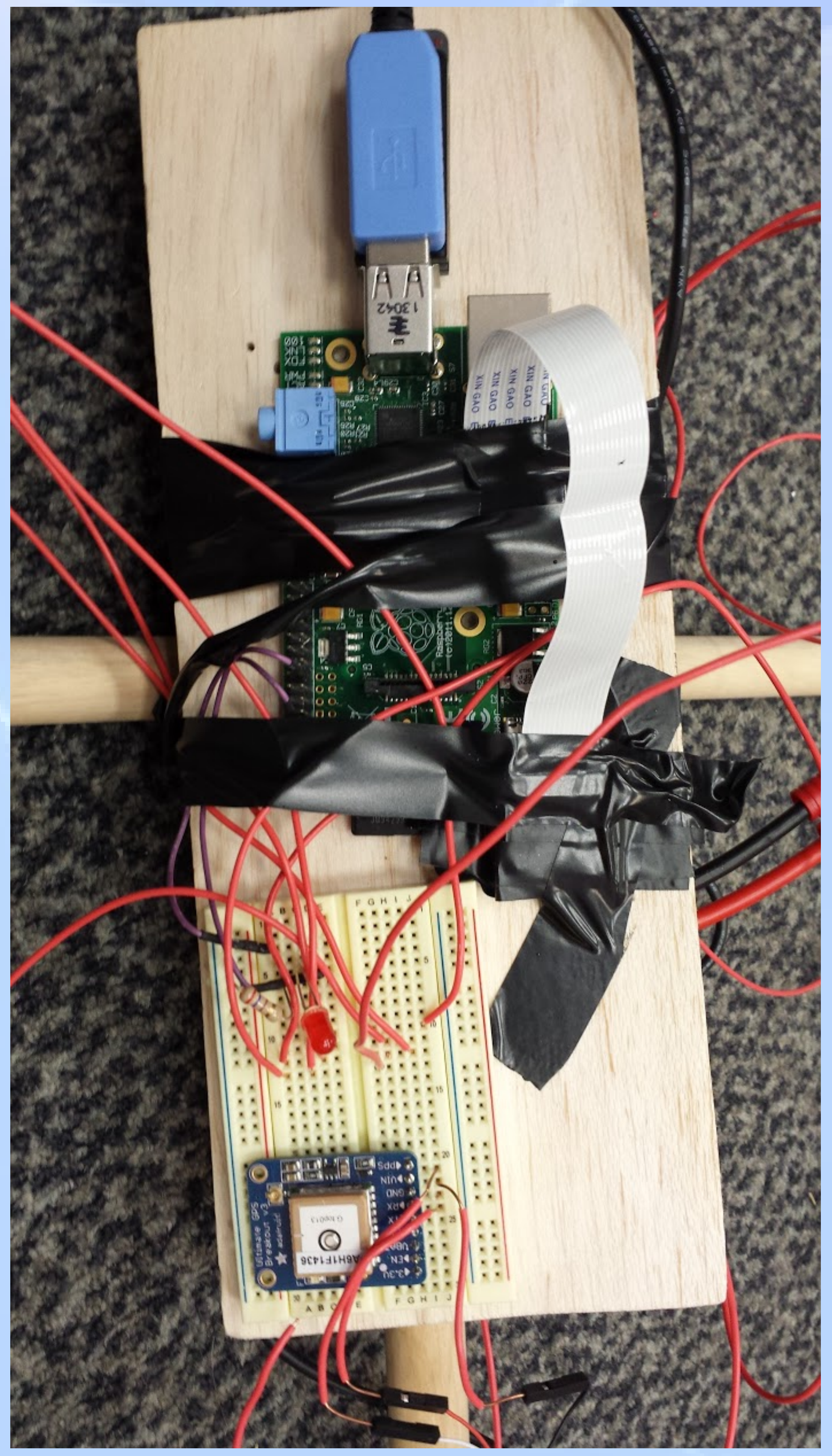


```
for pixel in pixelArray:
    differenceFromTarget = differenceBetween(pixel, targetColor)

    if differenceFromTarget < Imaging.FUZZINESS:
        gpsCoordinates = getLatLong(pixel)
        matchingCoordinates.append(gpsCoordinates)

        print "Matching pixel #%s with diffence of %s" % (matchingNum, differenceFromTarget,)
        matchingNum += 1
```

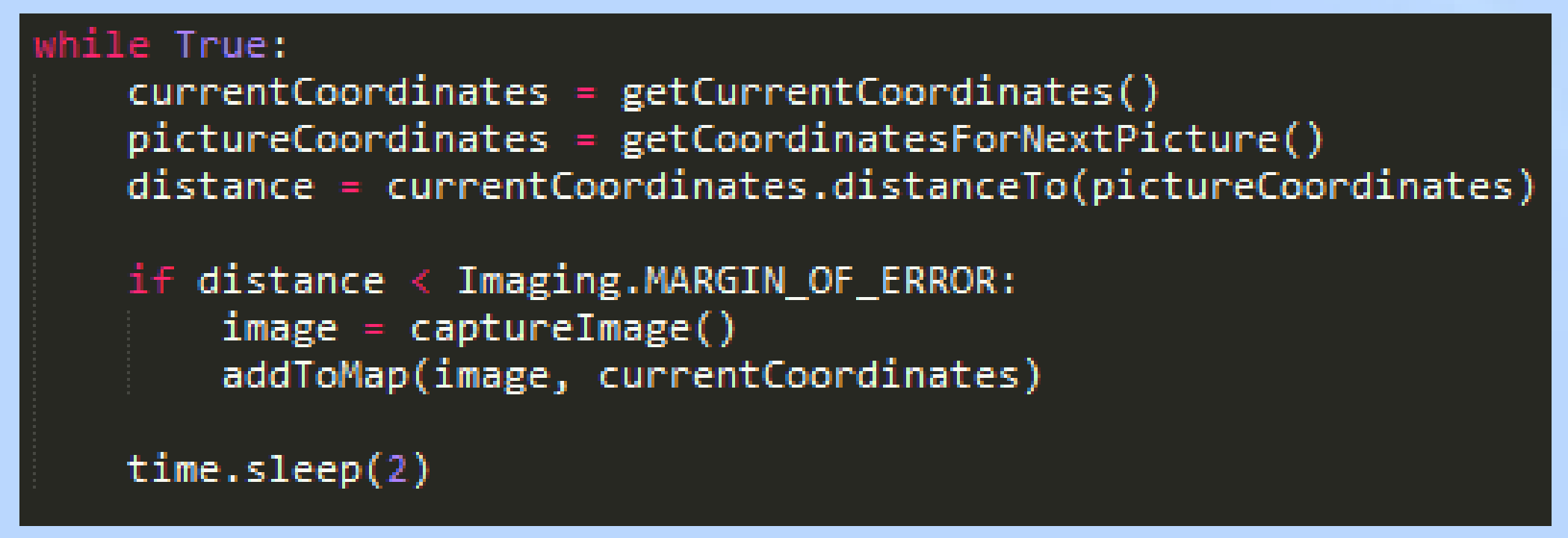
Code to search the image for matching pixels and keep track of their latitude and longitude

RESULTS

This project was ultimately successful in creating a quadcopter with an onboard computer system to image a given area and find the latitude and longitude of points of interest selected based on matching a given color. It was also successful in creating a Raspberry Pi module that can be fitted to any flying device to image and analyze a swath of land. The created system is relatively low cost and provides good accuracy over areas the size of a football field or smaller.



Quadcopter Control Circuitry



```
while True:
    currentCoordinates = getCurrentCoordinates()
    pictureCoordinates = getCoordinatesForNextPicture()
    distance = currentCoordinates.distanceTo(pictureCoordinates)

    if distance < Imaging.MARGIN_OF_ERROR:
        image = captureImage()
        addToMap(image, currentCoordinates)

    time.sleep(2)
```

Part of thread that automatically captures images at the specified GPS coordinates

LIMITATIONS

- The motors of the quadcopter drain the battery quickly, limiting flight time to approximately seven minutes. This limits the size of the area the device can scan.
- The software calculates latitude and longitude of pixels within the image by interpolating the latitude and longitude of the bounds of the image. This fails to account for the curvature of the earth and introduces some error, though this is negligible for distances capable of being imaged by this device due to battery limitations.
- The Raspberry Pi is small, inexpensive, and doesn't use much power, but it also lacks computing power. The Raspberry Pi can be slow to tell the quadcopter where to fly when processing an image, and can pause in the air for a few seconds, taking away from the limited flight time.
- Some inefficiency is introduced by relying on the Python interpreter for the program that controls the imaging and quadcopter.
- The quadcopter must maintain the same altitude throughout the flight and does not have any ability to avoid obstacles. The need to take all pictures from the same altitude comes from an assumption that the ground is level and at a set distance away to calculate the latitude and longitude of points on the ground.

FUTURE WORK

- Rewrite the control script in a compiled language to remove the computational costs associated with using Python.
- Replace the current battery with a larger and safer battery for longer flight times and safer charging.
- Calculate latitude and longitude of points with a model that accounts for the curvature of the earth and doesn't become inaccurate on larger scales.
- Add in obstacle avoidance when flying around to take pictures
- Use a LIDAR system to map the topography of the ground, removing the need to take pictures of flat ground at a predetermined altitude.
- Add more advanced image recognition, beyond searching for one color

REFERENCES

- https://www.cs.bath.ac.uk/brown/papers/ijcv2007.pdf
- http://www.cs.bath.ac.uk/brown/autostitch/autostitch.html
- http://www.ipni.net/publication/ssmg.nsf/0/0BDF314BF75B9FC1852579E5007691F9/$FILE/SSMG-11.pdf