Advanced Mechatronics: AR Parrot Drone Control Charging Platform

Engineering Team Members:

Ashwin Raj Kumar
Feng Wu
Henry M. Clever

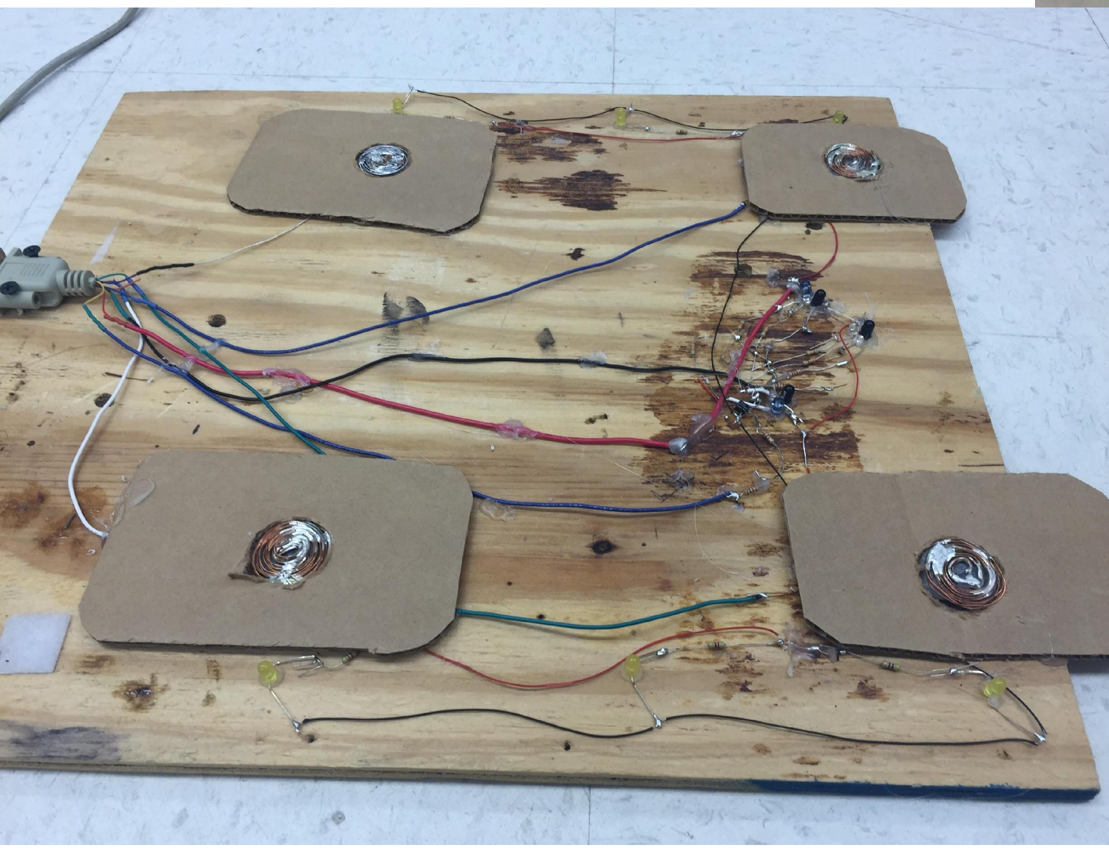
Advanced Mechatronics: Project Plan

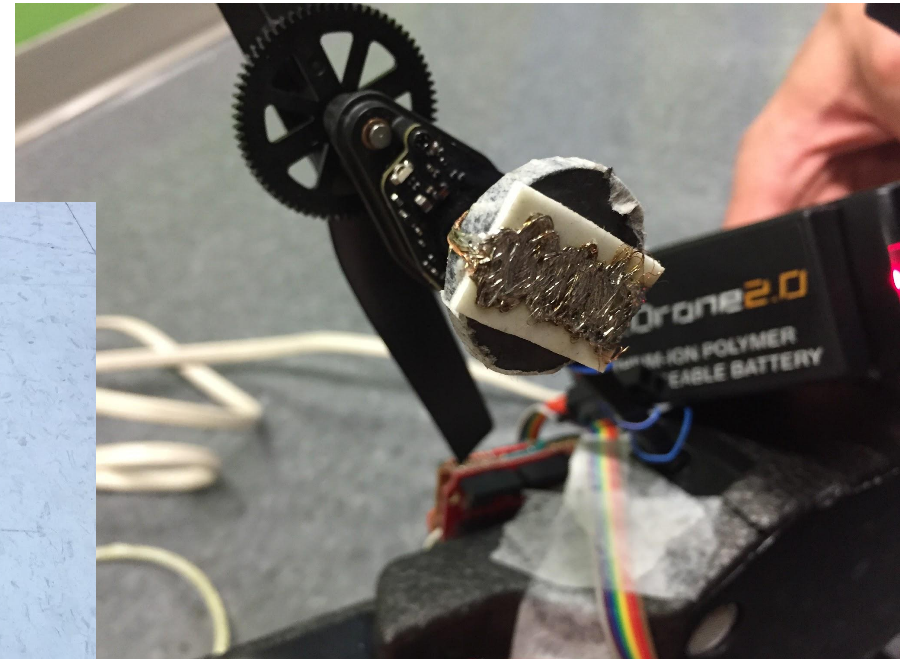
Phase 1: Design testing platform

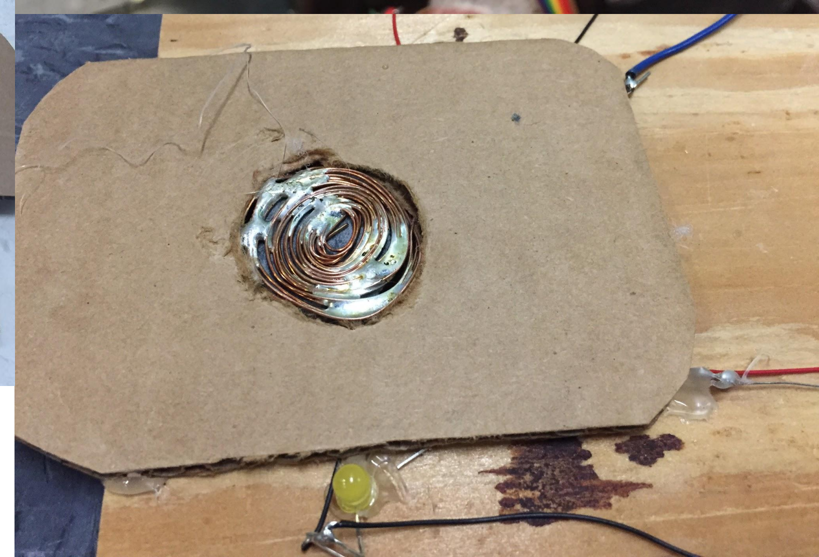
Phase 2: Automated landing sequence

Phase 3: Battery charging station + optimum control performance

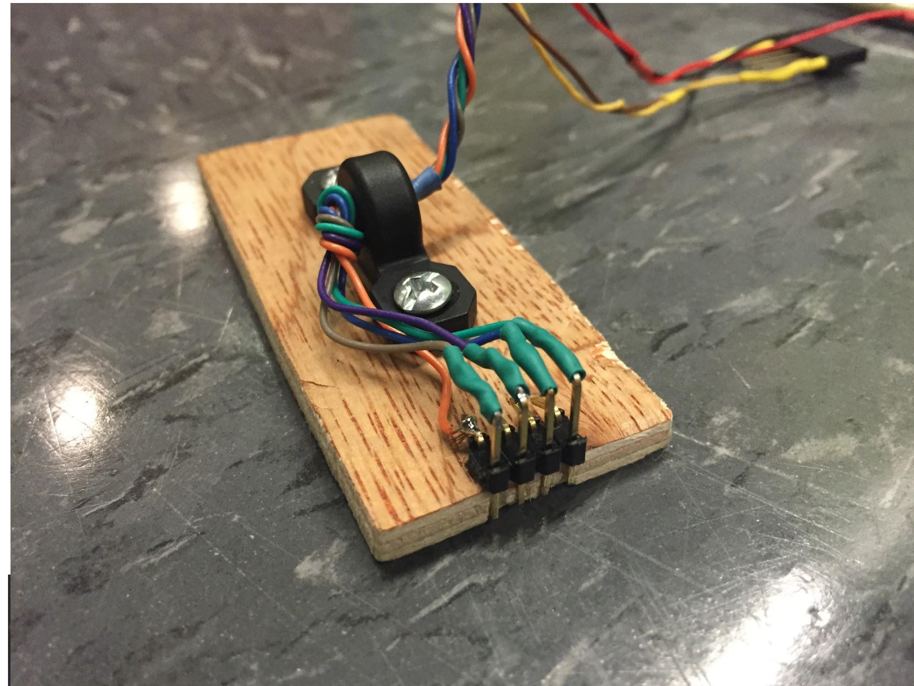
Landing Pad: Charging Wire

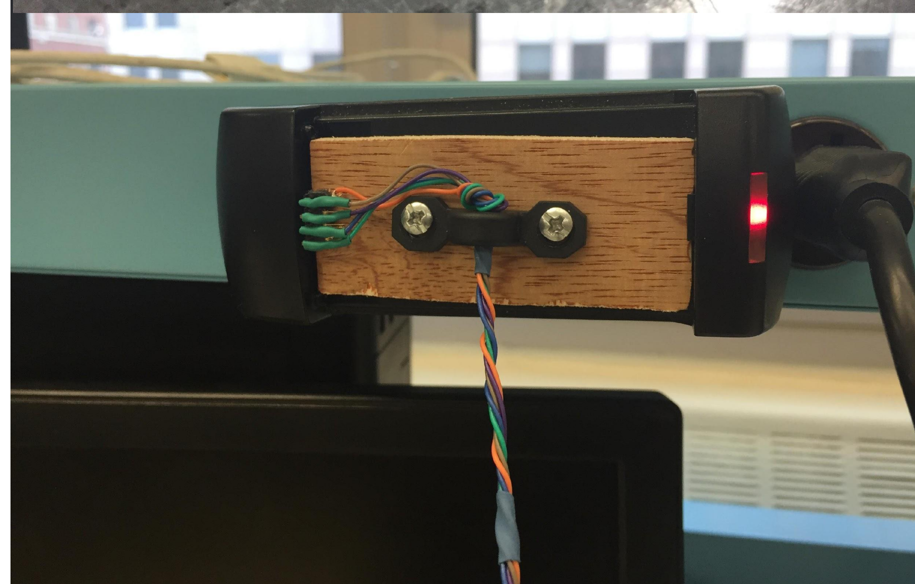




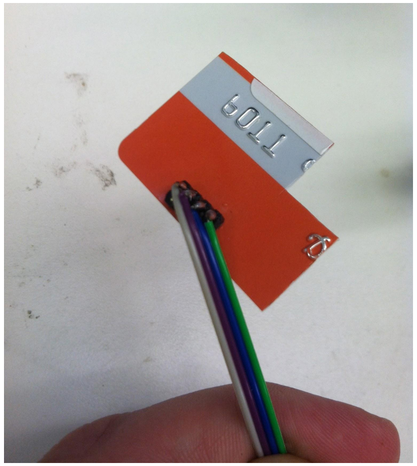


Charger Adaptor

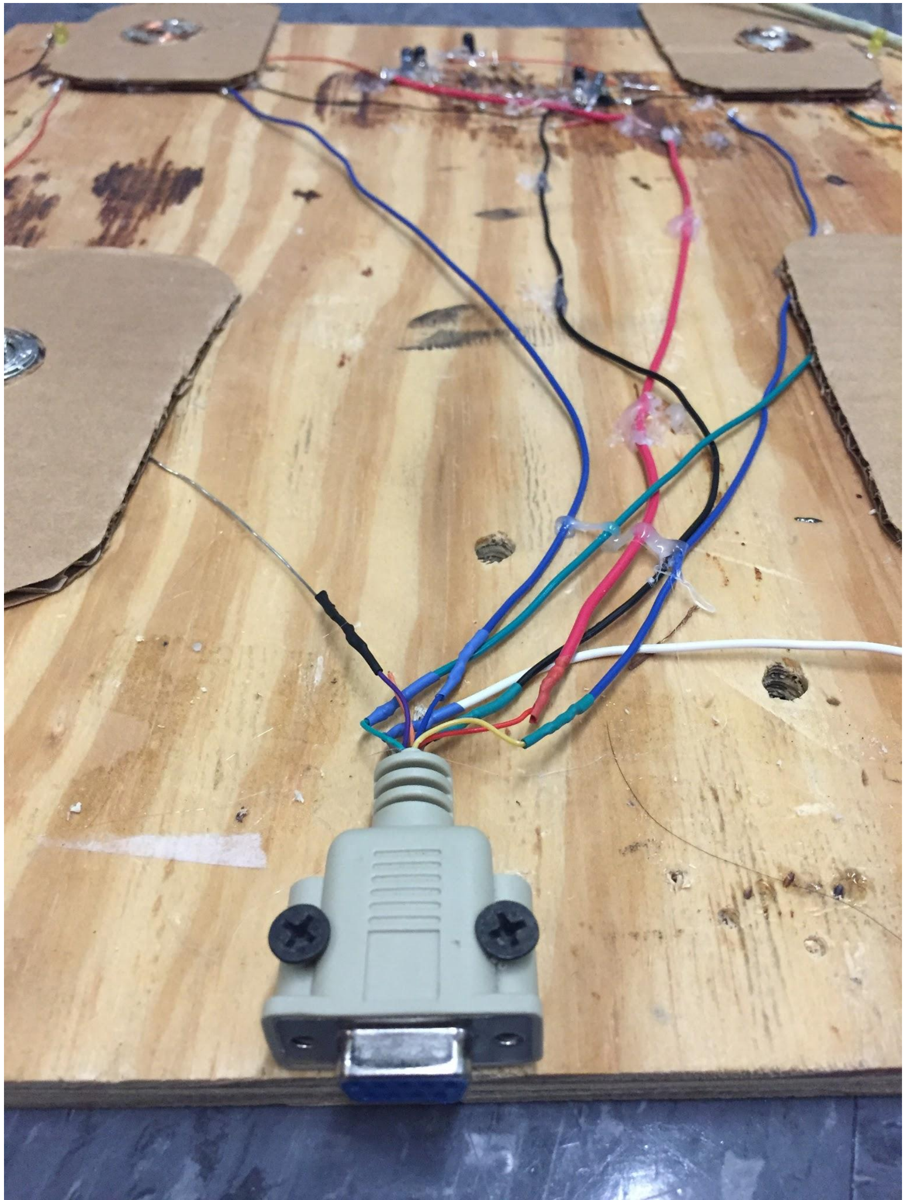


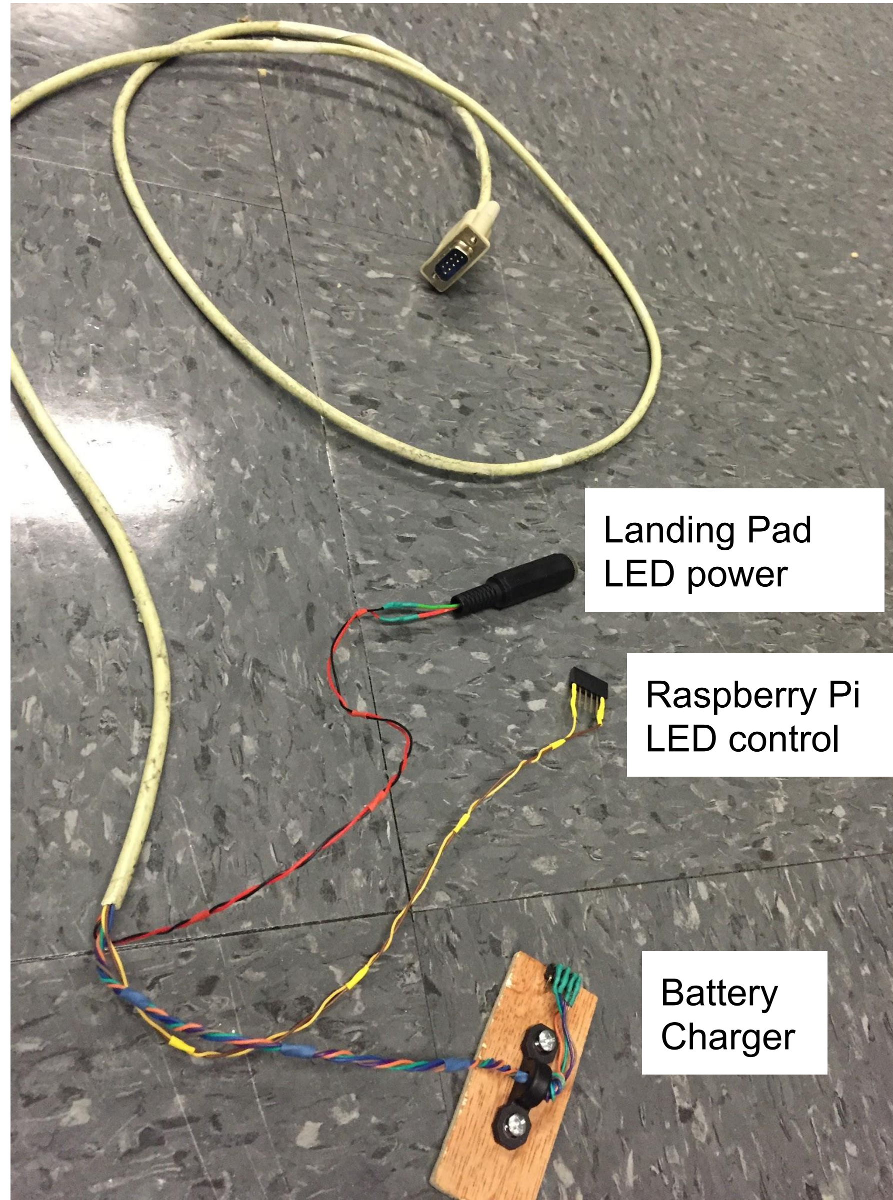


Battery Adaptor

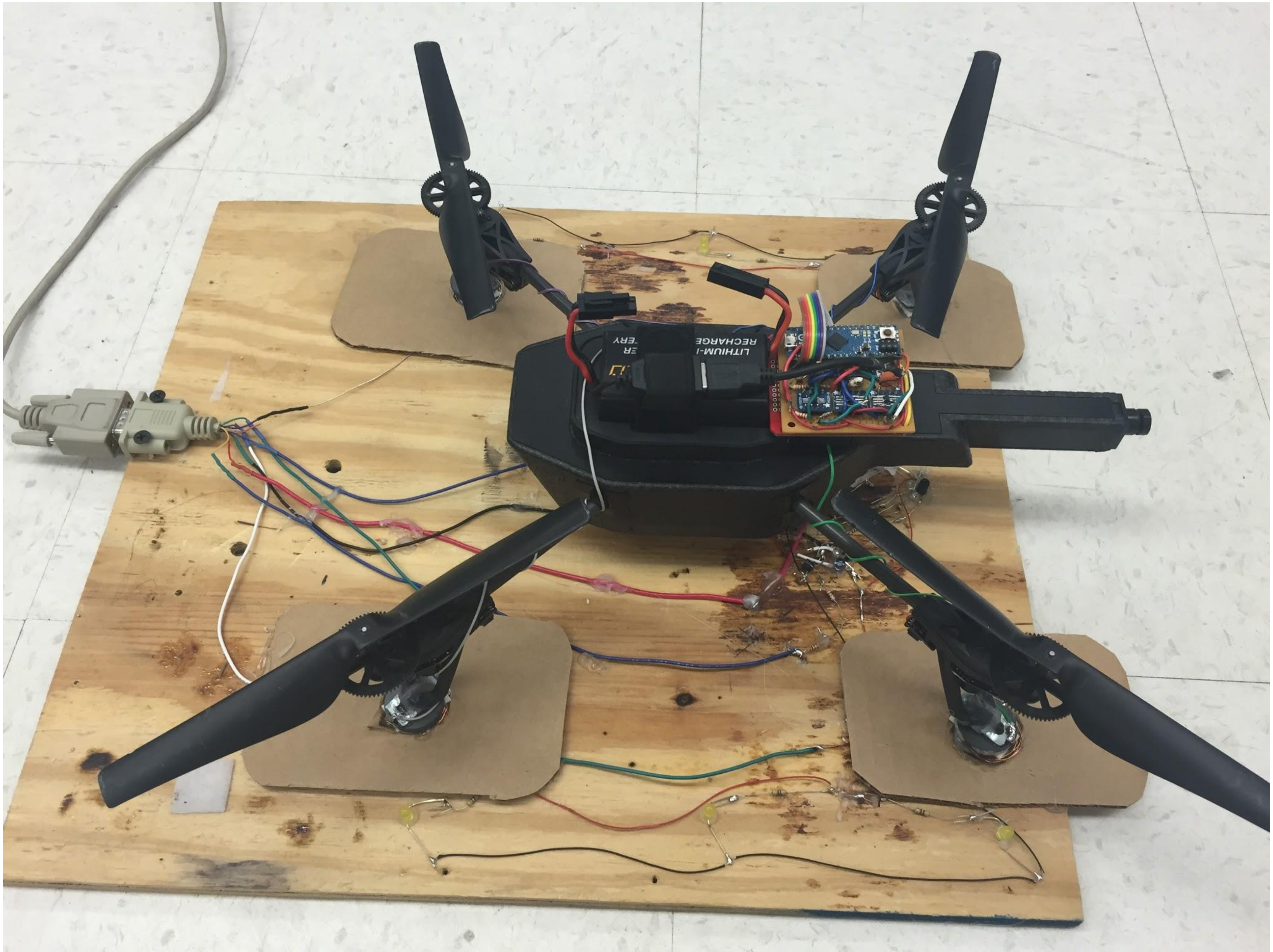


General Hardware Improvements

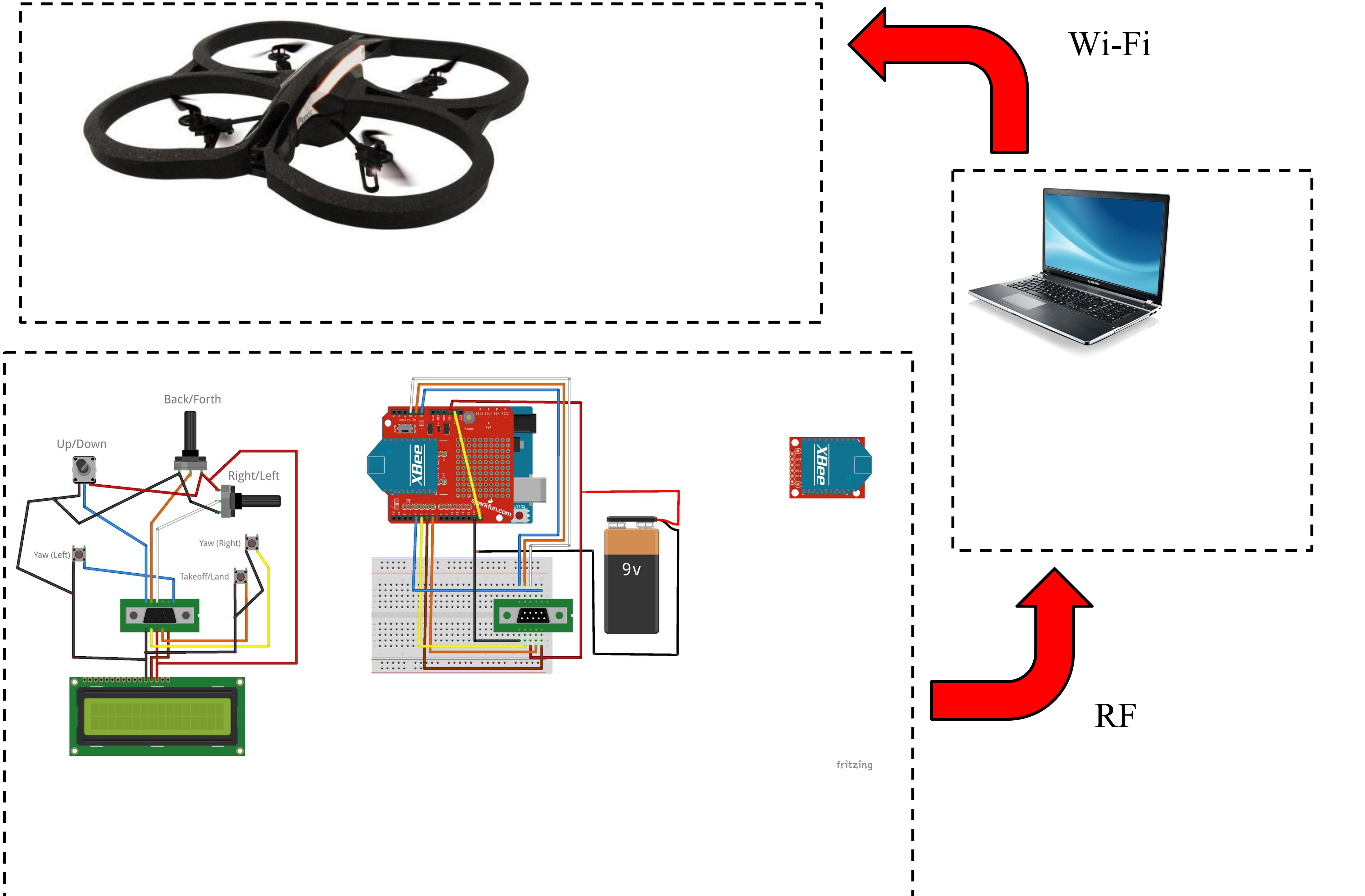




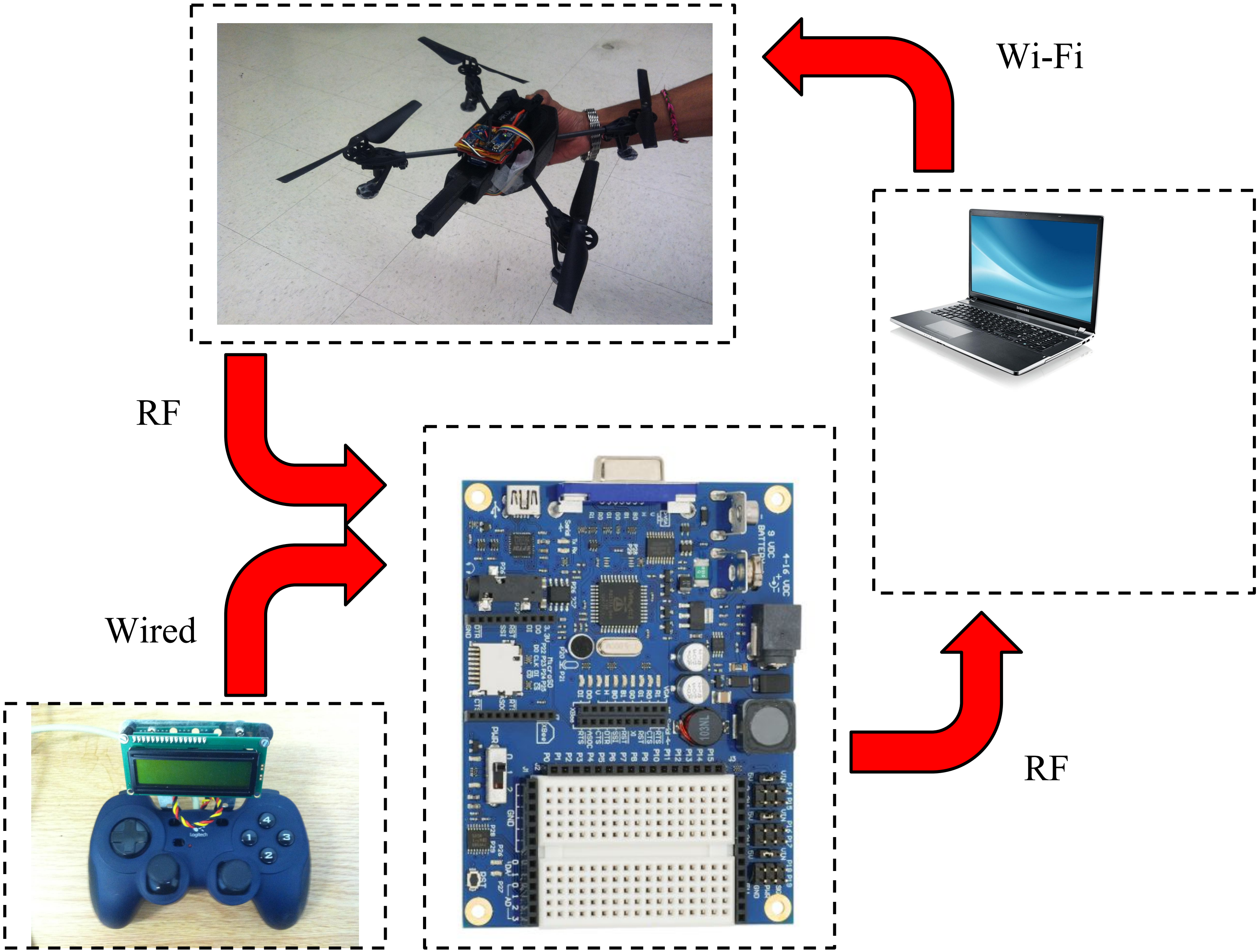
Overview



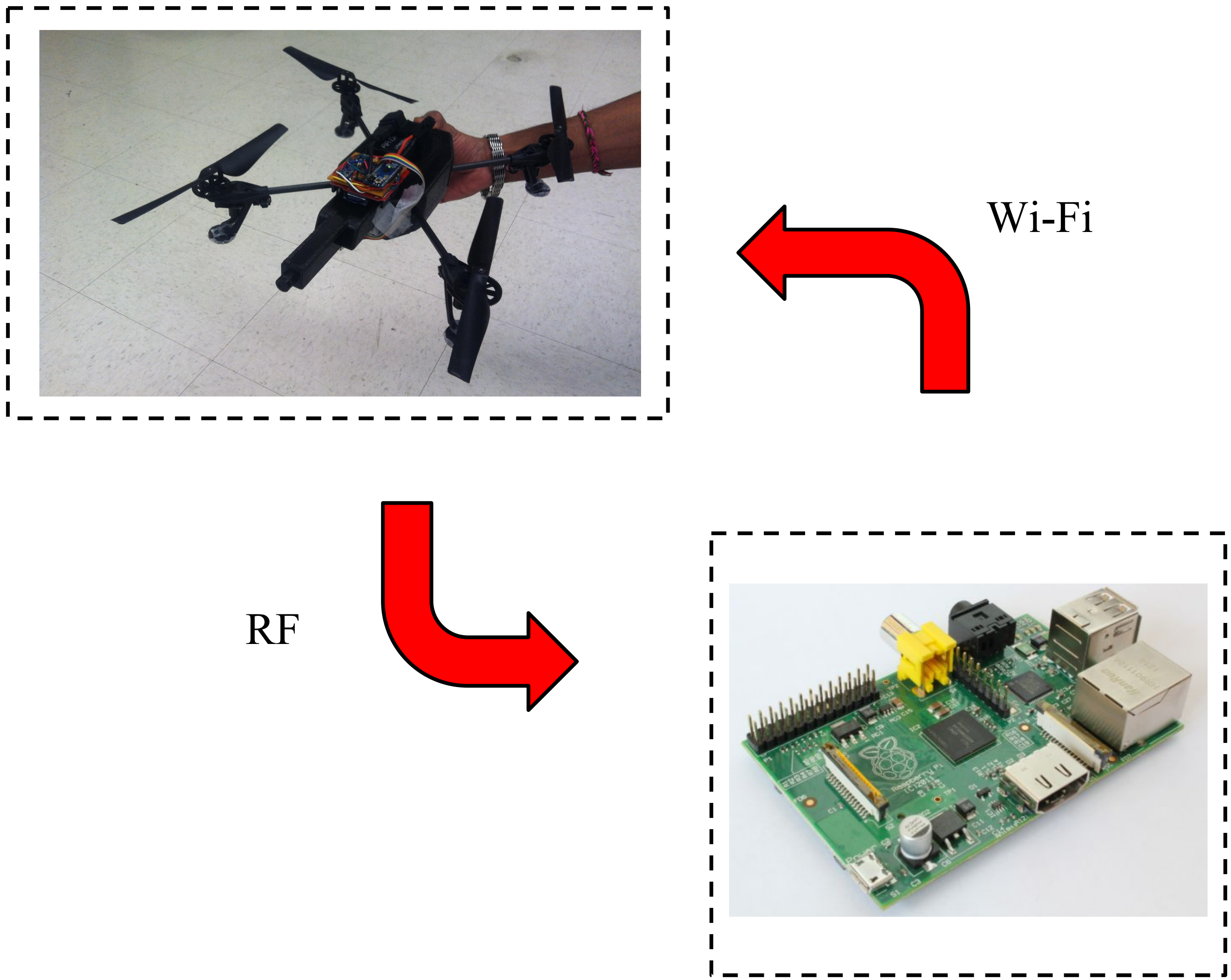
AR Drone System Schematic: Phase 1



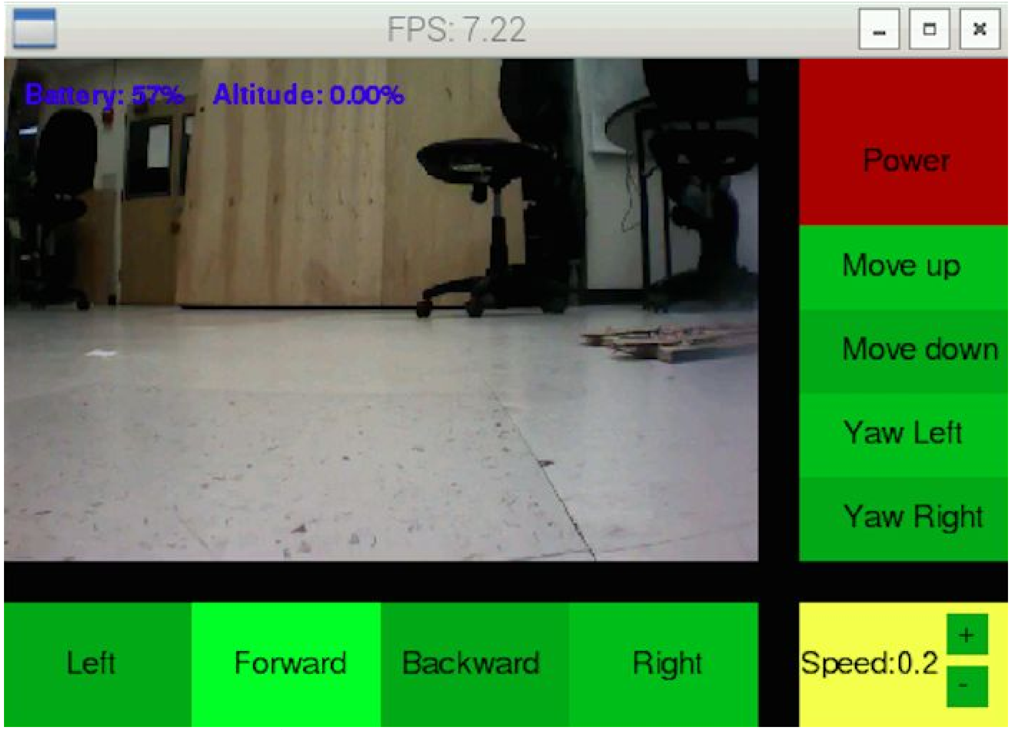
AR Drone System Schematic: Phase 2



AR Drone System Schematic: Phase 3



Codes for GUI Button

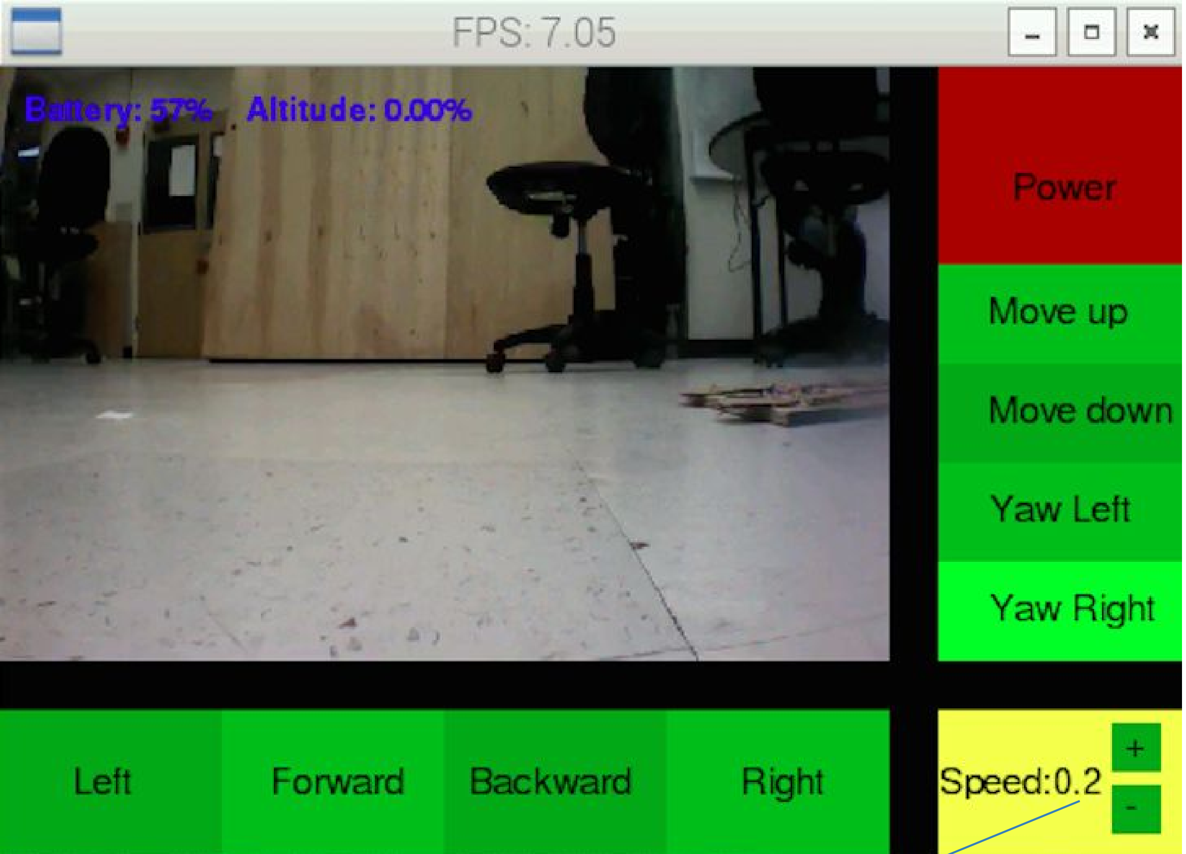



```
import pygame
```

```
def button(x,y,w,h,c1,c2):
    mouse1=pygame.mouse.get_pos()
    if x+w>mouse1[0]>x and y+h>mouse1[1]>y:
        pygame.draw.rect(screen, c1, (x,y,w,h))
    else:
        pygame.draw.rect(screen, c2, (x,y,w,h))
```

```
#forward button
button(90,260,90,60,green_bright,green2)
screen.blit(myfont.render("Forward",1,black),(110,280))
```

Codes for Variable Print




```
import pygame
```

```
speed_display=drone.speed
s="Speed:"+str(speed_display)
screen.blit(myfont.render(str(s),1,black),(380,280))
```

Codes for Button Control


```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        running = False
    elif event.type == pygame.KEYUP or event.type == pygame.MOUSEBUTTONUP:
        drone.hover()
        pygame.draw.rect(screen,black,(0,240,360,20))
        pygame.draw.rect(screen,black,(360,0,20,240))
    elif event.type == pygame.MOUSEBUTTONDOWN:
        if event.button == 1:
            #left
            if 90>mouse[0]>0 and 320>mouse[1]>260:
                print 'Left'
                drone.move_left()
                pygame.draw.rect(screen, yellow_bright, (0,240,90,20))
            #forward
            elif 180>mouse[0]>90 and 320>mouse[1]>260:
                print 'Forward'
                drone.move_forward()
                pygame.draw.rect(screen, yellow_bright, (90,240,90,20))
```

Communication between

```
t1 = Thread(target = manualControl)
t2 = Thread(target = automaticControl)
t3 = Thread(target = display)
if __name__ == '__main__':
     t1.start()
     t2.start()
     t3.start()

def automaticControl():
    ser = serial.Serial('/dev/ttyUSB0', 9600)
    while True:
        #while (ser.inWaiting==0):
        #    pass
        if (ser.inWaiting!=0):
            incoming = ser.readline().strip().strip('\x00')
            data=incoming.split()
            if len(data)==8:
                X1=int(data[0],base=10)
                Y1=int(data[1],base=10)
                X2=int(data[2],base=10)
                Y2=int(data[3],base=10)
                X3=int(data[4],base=10)
                Y3=int(data[5],base=10)
```

Drone Autonomous Landing

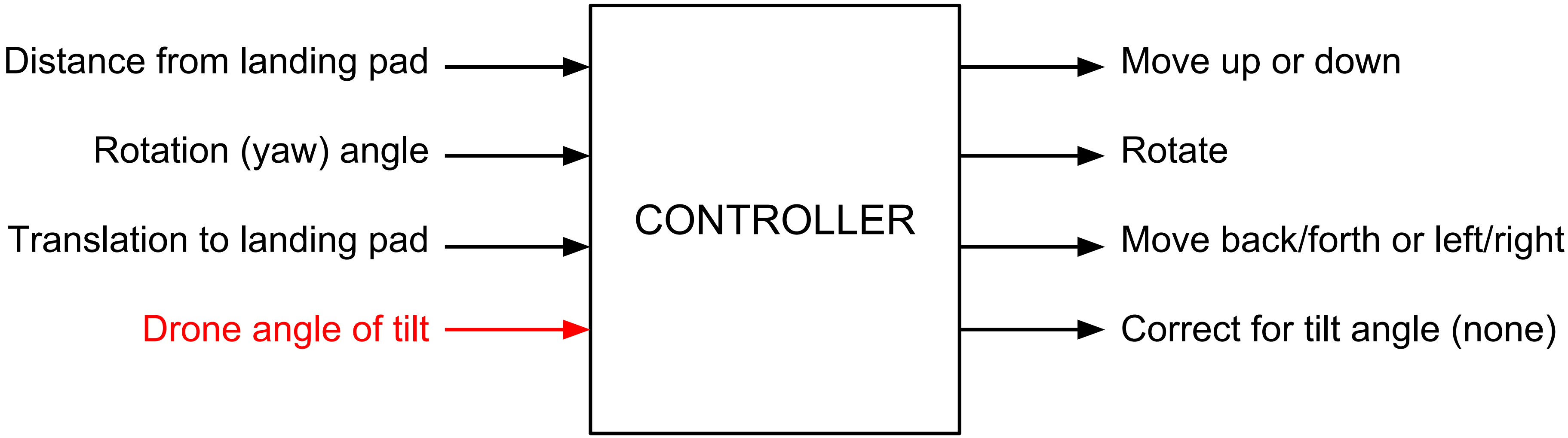
```
if(x1a<1023 and x2a<1023 and x3a<1023): # if all leds are
    GPIO.output(25,GPIO.HIGH)
    time.sleep(0.01)    #pause for 10 ms
    m1=(x3a-x2a)/(y2a-y3a)
    m2=(500-x1a)/(360-y1a)
    t=(m1-m2)/(1+(m1*m2))
    drone.speed = 0.1
    #Automatic Landing control
    if((x1a-500)>50):
        drone.move_left()           #go left
        if((360-y1a)>50):
            drone.move_forward()    #go forward
        elif((y1a-360)>50):
            drone.move_backward()   #go back
    elif((500-x1a)>50):
        drone.move_right()          #go right
        if((360-y1a)>50):
            drone.move_forward()    #go forward
        elif((y1a-360)>50):
            drone.move_backward()   #go back
    elif((360-y1a)>50):
        drone.move_forward()         #go forward
    elif((y1a-360)>50):
        drone.move_backward()        #go back
    elif(x1a>450 and x1a<550 and y1a>310 and y1a<410):
        #red point 1 is close to center
        #x1a found
        if(d1a>=220):       #height control, land if close
            drone.land()             #land
        elif (d1a<220):     #d3a = 368   d1a=400
            drone.move_down()        #lower drone
    drone.hover()
```

Consolidation

- Previous control system used the following hardware:
  - Wii camera + Arduino Micro + XBee for feedback
  - Propeller to use parallel programmed cogs for feedback modification and control of output plus hand controller manual control system
  - Manual joystick controller
  - Computer + XBee dongle with Processing code to communicate control feedback to Drone via wifi
- Current control system uses the following hardware:
  - Wii camera + Arduino Micro + XBee for feedback
  - Raspberry Pi for multithreading code to run GUI plus conversion of control feedback for automatic control
- This consolidation makes the system most suitable for users with disabilities who cannot use a manual joystick for normal operation
- Additionally, it removes the necessity of an extra computer operating system and propeller processor
  - Much easier to set up and move around
  - Fewer hardware parts reduces the possibility of problems with the system due to bad wiring connections

Results

- We have developed an autonomous landing pad that has capabilities to steer the drone and land it on the magnets of the landing pad
  - The Raspberry Pi incorporates all previous controls of propeller and processing into a single unit
  - With a good hand GUI, Raspberry Pi takes commands from manual control GUI and automatic Wii feedback to land the drone accurately.
- Our automatic landing controller is a multiple input multiple output (MIMO) system.
  - Angle of tilt still not accounted for: causes instability
  - Develop and implement linear quadratic regulator (LQR)



Control System: Major Issue

- Due to the difference in processing RAM of the R-Pi vs. a conventional computer, the R-Pi code crashes unexpectedly when running
    - Previously developed Linux libraries were used to link our python code with AR Drone wifi
    - We speculate these libraries are designed for operating systems with sufficient processing power to run the Drone camera and control system simultaneously
- But then a dilemma arises: How can the smartphone app work so well to control the drone on an even smaller OS than R-Pi?
    - More detective work is needed to modify the C libraries linking high level control commands to AR Drone wifi so that R-Pi can run them more efficiently: a non-trivial software problem.
- Although the system does work without error a good percentage of the time, this problem must be fixed for reliable use.

Results-2

Future Improvements

- Because of many crashes in testing the AR Drone, the blades are damaged and need replacement. There may be additional damages as well.
  - Solution: Buy a new AR Drone
- The current control system has not been tested enough to optimize the current control configuration
  - Solution: Perform more testing
- The electronics onboard AR Drone are off center from the COM and cause possible drift.
  - Solution: Modify electronics to fit in the middle
- MIMO systems perform better with a linear quadratic regulator (LQR)
  - Solution: Implement LQR with control scheme
- There are 3 inputs and 4 outputs on the MIMO loop
  - Solution: Use another sensor to monitor and correct for tilt angle