Locorobo/Drone Interface

Matt Fairfield, Chris Banksi, William Keeley

March 15, 2016

Contents

1 Required Files

HCD.zip (arduino library)
Flybot_ARDUINO.ino
LocoRobo (python library)
tkinter (python library)
LocoDrone_GUI.py

–need .gif files in /pics/ folder for GUI–
center.gif
down.gif
up.gif
left.gif
right.gif

2 Quick Start Guide

2.1 First time Setup

1. Add HCD.zip to the Arduino libraries. (Must use my modified HCD.zip for proper operation)
2. Install Sensor UDP app on android device here
3. Install LocoRobo and tkinter libraries in python.

2.2 Drone Operation Setup

1. Program the Arduino with Flybot_ARDUINO.ino file.
2. Ensure your phone and computer is on the same Wi-Fi network.
3. Open Sensor UDP app and input your IP address and an available port number.
    a. Enable Orientation to send pitch and yaw data.
4. Mount LocoRobo Bluetooth dongle to computer and power up your robot.
5. In Python code LocoDrone_GUI.py check/change necessary variables.
    a. COM ports for Arduino and Bluetooth dongle
    b. LocoRobo robots name
    c. Current IP address and Socket Port number (Must match Sensor UDP app)
6. Connect drone battery.
7. Run Python code

2.3 Drone GUI Operation

1. Must connect all three connections to enable all other GUI buttons.
    a. Connect Phone Button establishes connection with SensorUDP app.i.

Robot will play disconnect tones if phone connection is lost.

b. Connect Robot Button establishes connection with LocoRobo robot.

c. Connect Drone Button establishes connection with drone.

2. Fly Drone Button enables drone flying control from SensorUDP app.

3. Follow Robot Button enables LocoRobo random walk and makes drone mimic robot moves.

a. MAKE SURE ROBOT IS ON THE FLOOR BEFORE BUTTON IS PRESSED.

4. Stop Flight Button disables drone flying control and/or robot following routine.

a. Does not affect the drones throttle.

5. Throttle Off Button sets throttle payload to 0 (0-255)

6. Throttle Launch Button sets throttle payload to 200 (0-255)

7. Freeze Throttle Button freezes current throttle payload set by Throttle Control.

8. Throttle Control Button changes the drones throttle payload based on robots angle.

a. ROBOT SHOULD BE PLACED VERTICALLY BEFORE BUTTON IS PRESSED.

9. Trim Controls increment/decrement pitch and yaw trims by 1 per click.

10. Motor Controls basic flying controls

a. Increment/decrement pitch and yaw by 5 per click

b. Return pitch and yaw to center.

11. Shutdown Button closes all connections and exits program gracefully.

3 Python Functions

The purpose of this section is to explain in detail the python interface. This is what interfaces with the arduino before sending commands to the drone, and sends commands directly to the Locorobo ground robot. GUI elements are self explanatory and will not be covered. Note that any functions that begin with an underscore (ex: _init_) are meant to be ran in their own threads to prevent blocking other loops.

3.1 $\_init\_(self, parent)$

Performs initialization and setup of various parameters of the GUI, robot state, phone app, serial connection, and drone parameters. Most variables are set to $false$, *none*, or 0, with the exception of MAX_ANGLE and *total_connections*. MAX_ANGLE will change the limit (in degrees) on how extreme the drone can be tilted when using the phone app to fly. The default is 75°. *total_connections* keeps track of whether all three required devices (phone app, robot, and drone) are connected so that the GUI can be unlocked as soon as the required connections are completed. This value should stay at 3.

3.2 $CONNECT\_PHONE(self)$

Connects to the phone app via UDP. A timeout is set to 20 seconds, at which point the connection attempt will terminate and alert the user. If the connection is successful, $\_phone\_data(self)$ will be called in its own thread and *total_connections* is incremented.

3.3 $CONNECT\_ROBOT(self)$

Establishes a connection to a Locorobo ground robot. The global variable *name* is used to determine which robot is being used. Otherwise it will hijack the first robot that is sees. This function also initializes the motors, accelerometer, and ultrasonic sensor on the robot. If the connection is successful, *total_connections* is incremented, else it will timeout and ask the user to try again.

3.4 $CONNECT\_DRONE(self)$

Connects to the drone via a serial port to the arduino by writing a $'0'(utf-8)$. The timeout is set to 8 seconds. If successful, the drone will send an $'ACK'$ to verify the connection. **Note: MUST HAVE MODIFIED HCD.ZIP ADDED TO ARDUINO LIBRARY TO RECIEVE ACK'S.** Once the $'ACK'$ is recieved, *total_connections* in incremented.

3.5 $SHUTDOWN(self)$

Provides a polite way of closing out the connections to the robot, drone, and phone app. Loops are set to $false$ to make them exit, serial connection is flushed and closed, robot motors are disabled, and the user is informed that the devices have all be disconnected.

3.6 $\_phone\_data(self)$

Starts the *while* loop for relaying data from the phone app in 96 byte packets. This packet contain instructions for orientation, which is the combined pitch and yaw values. The other parameters such as acceleration, gravity, ambient light, etc, are all sent, but not used by the drone. It currently does not implement throttle control, but can be modified to include this feature if desired. If the connection to the phone is lost, the ground robot emits a tone, and the drone is commanded to level out to help prevent a collision.

3.7 $\_fly(self)$

Initiates the while loop that recieves data from the phone app. This data is send to the arduino via the serial port to control the drone's pitch and yaw. A moving average is used to smooth out the commands and eliminate and spikes or other data outliers. The value NUM_POINTS sets the sensitivity of the moving average, initially set at 4. The larger this number, the smoother the

flight controls will be, but at the cost of responsiveness. Lower numbers will make the drone much more sensitive to inputs, and will be increasingly difficult to control. There are three levels of directional control in each direction based on the angle on the phone: low, medium, and high. For example, if the pitch angle is between than $55°$ and $MAX\_ANGLE - 10$, the command for *pitch medium* will be transmitted to the arduino, to then be passed along to the drone. All characters must be sent as type $'UTF - 8'$ for the arduino to accept them.

3.8 $\_follow(self)$

Kicks off a new thread for the ground robot's $\_random\_walk()$ function and starts the *while* loop for the drone automated follow routine. If the robot is in state $'1'$, then the drone is commanded to pitch forward based on the value set in the arduino code. If the robot state is $'2'$, that means it is performing a turn. The amount of time it spends turning is mirrored for the drone. Therefore, for the drone and robot to sync up, the value for the drone yaw has to be adjusted in the arduino code to get the desired effect. Finally, any state other than $'1'$ or $'2'$ will command the drone to level out.

3.9 $\_throttle\_control(self)$

Starts the *while* loop for the drone's throttle control. For this feature to be accessible, a second ground robot must be used. To control the throttle, stand the second robot on its back end with the bottom facing away from the user. Tilt it forward slowly to increment the throttle and gain altitude. Tilt backwards to decrement the throttle. The step size for the throttle changes can be changed in the arduino code to adjust the sensitivity. Note that this is not required if flying the done inside, as it will ride along the ceiling at full throttle by default.

3.10 $\_random\_walk(self)$

Before implementing this command, the drone should be centered over the ground robot and oriented in the same direction. If indoors, starting the drone on the ceiling is ideal to prevent it from bouncing off at full throttle. This function commands the ground robot to perform a predetermined number of random walks based on the expression $for\ i\ in\ range(\#)$, where $'\#'$ is the number of steps. In each step, the robot will move forward for a random distance, based on the multiplier for random.random(). Once that value is reached, or the ultrasonic sensor detects an obstacle, the robot will perform a random turn and repeat the process. The phase of the robot is tracked so that the drone will know how long to move forward or turn the correct direction. Once the total number of steps is reached, the thread is terminated, and the drone is set to level flight. At this point the user can initiate another follow, free flight with the phone app, or disconnect.

4 Arduino Functions

4.1 *void inc*255(*int idx*)

Takes an index as an argument corresonding to a particular axis and increments the payload of hat axis by 1. This payload, when incremented in this way, has a hard limit of 255.

4.2 *void dec*255(*int idx*)

Takes an index as an argument corresonding to a particular axis and decrements the payload of hat axis by 1. This payload, when decremented in this way, has a hard limit of 0. Functions the same as dec 128().

4.3 *void inc*128(*int idx*)

Takes an index as an argument corresonding to a particular axis and increments the payload of hat axis by 1. This payload, when incremented in this way, has a hard limit of 128.

4.4 *void dec*128(*int idx*)

Takes an integer index as an argument corresonding to a particular axis and decrements the payload of hat axis by 1. This payload, when decremented in this way, has a hard limit of 0. Functions the same as dec255().

4.5 *void Change_Payload*(*int idx*, *int new_speed*, *float durr*)

By passing an integer index and integer speed(0-255), this will set that axis to a particular numerical setting. A float is passed to change the duration of the command before the drone returns to the preset neutral setting.

4.6 *void Change_Payload*(*int idx*, *int new_speed*)

By passing an integer index and integer speed(0-255), this will set that axis to a particular numerical setting.

4.7 *void Make_Center*(*int cmd*)

Takes an integer cmd corresponding to a particular axis and sets it to center. It does this by simply calling Change_Payload and sends the axis index and 127 for the setting.

4.8 *void loop*()

Main loop executed by the arduino. It continually checks the serial port for new information in the form of an unsigned char. It uses this as a case statement to determine which function is called to issue a command to the drone.

5 Drone Commands

UTF-8 Character	Issued Command
0	bind drone
Throttle Controls	
9	throttle shutoff
8	throttle launch
7	throttle increment
6	throttle decrement
Centering	
5	center yaw and pitch
3	center yaw
2	center pitch
Timed Movements	
o	follow pitch
p	follow yaw
X	pitch forward HI
W	pitch forward MED
w	pitch forward LOW
x	pitch backward HI
S	pitch backward MED
s	pitch backward LOW
Z	yaw right HI
D	yaw right MED
d	yaw right LOW
z	yaw left HI
A	yaw left MED
a	yaw left LOW
Constant Movements	
i	pitch forward
k	pitch backward
j	yaw left
l	yaw right
m	roll left
,	roll right
Trim Adjustment	
t	pitch forward
g	pitch backward
f	yaw left
h	yaw right
v	roll left
b	roll right