FINAL PROJECT:

TRICOPTER & GUI

Class: ME-GY 6933 Advanced Mechatronics

Students: Sunglyoung Kim, Devin Braatz, Ezra Idy
Professor: Vikram Kapila

Due: 05/11/17

ABSTRACT:

The objective of the project is to design, create, fabricate, and test a tricopter that can be used for home surveillance system. A tri-copter is a aerial vehicle with three propellers. When working creating a tricopter the roll and pitch of the copter is similar to other drones, such as a quadcopter, however the yaw of a tricopter requires a yaw mechanism that can angle the back propellor. When designing the frame, the max weight of the copter was calculated to be less that 1.2kg. An Arduino Nano and a 9 DOF (Degrees of Freedom) IMU are used to control the flight of the tricopter. By using the IMU, a PID controller can be created to effectively control the roll, pitch, and yaw of the tricopter. Also connected to the tricopter is a Raspberry Pi that will allow the user to create flight logs as well as obtain constant visual feedback from the tricopter. The Arduino and the Raspberry Pi are in constant communication with each other and transfer data with each other. Live streaming video that attached on the tricopter can be recorded and stored on a web server.

BACKGROUND:

When designing a tricopter, the first step that needs to be completed is calculating the amount of thrust that each propeller can emit. From that value one can determine the weight and stability of the given tricopter. The thrust calculations depend on the propeller diameter, propellor pitch, and the rotational value of the motors. Using the equation for thrust shown in Figure 1 below, as well as the propeller inputs shown in Figure 2, one can obtain the static thrust that can be produced by each propeller. These values can be seen in Figure 2 below.

$$F = 1.225\frac{\pi(0.0254\cdot d)^2}{4}\left[\left(RPM_{prop}\cdot 0.0254\cdot pitch\cdot\frac{1min}{60sec}\right)^2 - \left(RPM_{prop}\cdot 0.0254\cdot pitch\cdot\frac{1min}{60sec}\right)V_0\right]\left(\frac{d}{3.29546\cdot pitch}\right)^{1.5}$$

Figure 1: Thrust Equation

	Propeller Inputs						
	diam, d (in):	10	pitch (in):	4.7	RPMs:	5700	
	x		y				
	Aircraft Airspeed, V0 (m/s)	Aircraft Airspeed, V0 (mph)	Dynamic Thrust, F (N)	Dynamic Thrust, F (g)	Dynamic Thrust, F (kg)	Dynamic Thrust, F (oz)	Dynamic Thrust, F (lb)
Static Thrust -->	0	0	4.14174073	422.1957	0.422195	14.89253	0.930781
all others are dynamic thrust	0.44704	1	3.97848286	405.5538	0.405553	14.30550	0.894092
	0.89408	2	3.81522500	388.9118	0.388911	13.71847	0.857402
	1.34112	3	3.65196713	372.2698	0.372269	13.13144	0.820713

Figure 2: Thrust Table

After calculating the thrust values, one must then calculate the yaw angle. Since there are three propellers, in order to compensate for the yaw movement one of the propellers must be adjusted. Figure 3(a) below displays the tilt equations and the tilt equations, while Figure 3(b) displays the degrees needed for the tilt with respect to the given RPM value. From Figure 3(b) one can see that the tilt angle will not exceed 3 degrees. It is this tilt angle that allows the tricopter to steady the yaw component of its flight.

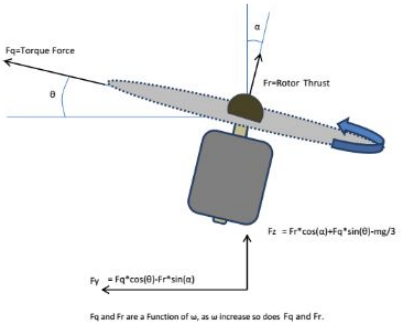

Figure 3(a): Tilt Equation

10x4.7							
RPM	CT	CP	T(N)	T(g)	Tan(alpha	Alpha(rad	Alpha(Degree)
2377	0.1059	0.0431	0.844008	86.03547	0.040445	0.040423	2.316091
2676	0.1079	0.0437	1.089898	111.1007	0.035751	0.035736	2.047525
2947	0.1079	0.0437	1.321825	134.7426	0.032464	0.032452	1.859377
3234	0.1104	0.0444	1.628701	166.0245	0.029376	0.029367	1.682632
3494	0.1117	0.045	1.923495	196.075	0.027237	0.02723	1.56016
3762	0.1143	0.046	2.281792	232.5985	0.02527	0.025265	1.447573
4029	0.1158	0.0466	2.651522	270.2877	0.023594	0.023589	1.351574
4319	0.1177	0.0474	3.096956	315.6938	0.022026	0.022022	1.261794
4590	0.12	0.0484	3.566143	363.5213	0.020757	0.020754	1.18913
4880	0.1223	0.0494	4.108264	418.7832	0.019552	0.01955	1.120122
5147	0.1237	0.05	4.622429	471.1956	0.018551	0.018549	1.062763
5417	0.1252	0.0508	5.1822	528.2569	0.017694	0.017692	1.013667
5715	0.1263	0.0513	5.818727	593.1425	0.016789	0.016787	0.961827
5960	0.1278	0.052	6.403473	652.7495	0.016127	0.016125	0.923908
6226	0.1286	0.0524	7.031555	716.7742	0.01546	0.015458	0.8857
6528	0.1299	0.0531	7.808393	795.9625	0.014792	0.014791	0.847449

Figure 3(b): Tilt Angle

In the figure below, one can see the two models of the tricopter frame used. The frame design on the left is the first design implemented for the frame. The frame was made from pieces of acrylic and the arms were 3D printed using ABS plastic. After running test and calculations on the first frame design, it was clear that the frame was not efficient and could be improved. After adjusting the propeller PID values it was clear that the flat triangular frame was producing a lot of drag, and that the space can be reduced. Thus the hexagon frame was constructed. In order to keep the mass of the tricopter at the center, the frame consists of multiple layers, allowing for a smaller frame and reducing the drag created by the propellers. Another way to reduce the unnecessary weight of the copter was by changing the frame material from acrylic to carbon fiber. With a carbon fiber frame, the tricopter frame can be stronger and lighter.

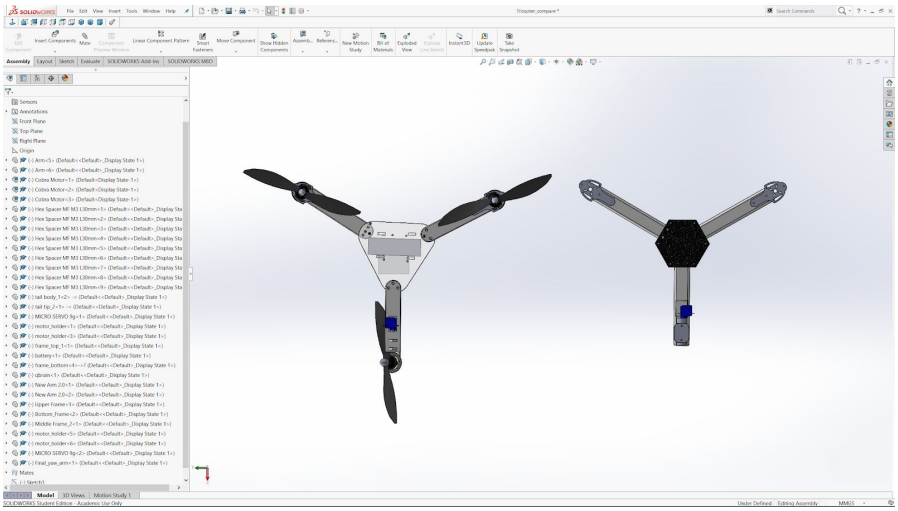

Figure 4: Frame design

The yaw mechanism is whole revised. The novelty of this yaw mechanism is manufacturing process. The whole rear arm is printed from 3D printed as a one piece. As Elite 3D printer at the makerspace can print with two materials, support material that soluble in an acid bath, and ABS for actual structure. Figure 5 shows final design of the yaw mechanism that attached on the rear arm.

Figure 5: yaw mechanism

In order to test the thrust values and create a PID controller, a test rig was built that can maximize the allowable freedom of the copter. One issue faced when testing on the ground is the propellers grounded effect. The grounded effect is due to the thrust bouncing off the ground and producing extra lift force. To eliminate the grounded effect produced, the test rig allows for the tricopter to be elevated. In order to test all roll, pitch, and yaw values the test rig consists of a ball in socket device. This allows for complete freedom in rotation and constrains the Z-axis, preventing it from flying. Once everything is set up and attached, the tricopter can be tested and values can be adjusted in order to maintain stability. Figure 6 below shows an image of the test rig that was constructed.

Figure 6: Test Rig

When designing a tricopter, safety is always the first priority. In order to ensure that the safety of those around the test rig two fail safes are put in place. The first act of safety is to ensure that the initial thrust isn't full power. Before the thrust can be adjusted the Arduino must read that 1100 RPM of thrust was given to the propellers. The second safety mechanism allows the user to cut the thrust by adjusting the joystick on the controller. This is put in place in case an immediate shutdown of the propellers are needed.

CODE:

The interface that controls the tricopter is broken up between two microcontrollers: the Arduino Nano and the Raspberry Pi. The Arduino Nano controls the thrust values of each propeller and the kinematics of the tricopter. The Raspberry Pi is used to send live stream video to the user as well as receive data from the Arduino Nano. The communication that is used between the Arduino Nano and the Raspberry Pi is Serial communication. Through the serial communication the IMU data is sent to the Raspberry Pi which is then sent to a web server, where it is displayed in a GUI along with the camera feedback. When using the PiCam with OpenCV it produces high delays and isn't good for live streaming. Instead the PiCam is sending its feedback directly to a server that can be accessed by a web browser.

Code 1 below displays an interrupt created using the Arduino interface. With the interrupt below, the Arduino reads the values given by the IMU every 250 ms. This interrupt is for controlling the propellers with the transmitter. Although 250 ms is not the fastest at which it can run, it is the most efficient. Due to the fact that the reaction time of a human is around this speed, any speed lower than 250 will not be observable. Instead, it will unnecessarily call the pulsein functions, the most time consuming part of the code.

```
void init_timer1_ovf_interrupt(void)
{
cli(); // disable all interrupts
TCCR1A = 0; // Clean the registers
TCCR1B = 0;
TCCR1B |= (1 << CS11); //
TCCR1B |= (1 << CS10); // Prescaler 1024
//divide timer1 timebase by 1024, overflow occurs every 250 ms
TIMSK1 |= (1 << TOIE1); //enable timer1 overflow interrupt
//TIMSK1 = 0x01; //enable timer1 overflow interrupt
asm("SEI"); //enable global interrupt
}
ISR(TIMER1_OVF_vect){
 receiver_flag=1; // Sets flag to get receiver values

}
```

Code 1: **Interrupt for transmitter**

```
void PID(float roll,float roll_prev, float pitch, float pitch_prev, float yaw, float yaw_prev){
  // Roll variables
 static float Ir_error;
  float kr=4.3;
  float ir=0.000;
  float dr= 0.1;
  float Pr_error;
  float Dr_error;
 // Roll PID
   Dr_error=roll_prev-roll;
   Pr_error=micro2degrees(desired_roll)-roll;
   Ir_error+= Pr_error*.01;
   adjust_roll= kr*Pr_error+ir*Ir_error+dr*Dr_error;
```

Code 2: PID

To obtain the given PID values that work best with the tricopter, the IMU is needed. From the IMU, one can obtain the Euler angles produced by the copter. Once the angles are given, it is possible to create a PID controller by playing with those numbers and multiplying by set constants. Code 2 above shows a snippet of how the PID values are calculated and adjusted. The PID values are calculated in the Arduino code and is then displayed on the web browser through the help of the serial communication with the Raspberry Pi. The flight control system of the tricopter is an example of a closed loop system, which can also be seen in the Code 2 above.

```
void get_receiver_values(void){
    int ch1, ch2, ch3, ch4;
// Measure transmitter values with a timeout of 40000 us
 ch1=  pulseIn(THROTTLE,HIGH,40000);
 ch2 = pulseIn(RUDDER, HIGH,40000);
 ch3 = pulseIn(ELEVATION, HIGH,40000);
 ch4 = pulseIn(AILERON, HIGH,40000);

 if (ch1 > 0)
  {Thrust = ch1;}
 if (ch2 > 0)
 {desired_yaw = ch2;}
 if (ch3 > 0)
  {desired_pitch = ch3;}
 if (ch4 > 0)
 {desired_roll = ch4;}
```

Code 3: Receiver values pulse in code

Code 3 above displays the receiver code that allows the propellers to be controlled by the transmitter. The transmitter originally sends a ppm signal to the receiver module. This ppm signal is split into six different channels. As shown above, the receiver code uses the pulseIn function to get the pulse widths of the various channels. For our purposes, four are used and they correspond to thrust, yaw, pitch, and roll. A timeout is used to minimize the time spent in the pulsein function. The pulse widths are then checked to see if they have valid values since in the case of a loose connection any garbage values of 0 would make the tricopter fall from the sky and crash.

```
def TCP_transfer():
    try:
        data_byte = conn.recv(1024)
        data_string=str(data_byte,'utf-8') # Turn byte string into utf-8 string

        try:        # Filters out blank space from pyserial
            roll,pitch, yaw= data_string.split(',')
            # Update global variables with float values
            rollf=float(roll)
            pitchf=float(pitch)
            yawf= float(yaw)
        except ValueError:
            print ("value error")

        print("Client Says:",roll,pitch,yaw, sep= " ")
        conn.sendall(b"Server Says:hi")

    except socket.error:
            print("Error Occured.")
```

Code 4: Python TCP server

The code above sets up the server side of a TCP connection between the raspberry pi and any other TCP compatible device. First, data is received and put into a 1024 byte buffer. This data comes in the form of a byte string, a new data type brought into python 3. This data type is not very clean so it is converted into utf-8 string format. Then the string is split into the roll and pitch values sent by the arduino to the raspberry pi. These are then turned into floats and sent to the global variables to be used by the GUI. Multiple try statements are used in the code for error handling. The first one detects if the socket crashed, mostly for debugging purposes. The second one handles incorrect values for the split method since occasionally raspberry will receive blank

values since it is just listening on the serial port and the arduino is only sending a periodic message. This also ensures accurate values are received.

The last part of the software was implementing a GUI. This was done by using PyQt5, a set of python bindings for Qt version 5. Qt is a desktop application software meant to streamline the creation of GUI and does a decent job of reducing the normally large amount of code associated with GUI development. Despite this, there was still a sizeable amount of setup code to get even the simple tricopter GUI up and running. The main reason was that we wanted the TCP server function to still run during the GUI which required the use of threading and timers. By using the proper libraries, we were able to successfully maintain our TCP server and update values accordingly. Finally, to make the GUI an overall package, it was arranged so that the web server hosted on the raspberry pi could be launched with a simple button press.

CONCLUSION:

The tricopter was successfully built, although there were many iterations made in the process. As the the built progressed, so did the difficulty in controlling the copter. After many tests, it is speculated that the disturbance in the PID control system comes from the tail motor. Due to the uniqueness of a tricopter, the tail motor is unlike that of another copter. Although the PID control needs more fine tuning, the live visual feed works perfectly. Since the copter was meant to be used for security purposes, the live feed is necessary for the tricopter design. With the live feed, it is possible to obtain all the tricopter information from one's desktop.

In the future, the PID controller can be tuned to produce better results. The frame can be redesigned in order to create better lift and maneuverability, in the air. The GUI will also go through a redesign, making it more user friendly and provide more information about the tricopter.