Power Line Tracking Unmanned Arial Vehicle (UAV)

M. Solilo and W. Doorsamy
Department of Electrical and Electronic Engineering Technology
Faculty of Engineering and Built Environment, University of Johannesburg,
Mcebisisolilo10@gmail.com; za.wesley.doorsamy@ieee.org

B.S. Paul
Institute for Intelligent Systems,
University of Johannesburg,
babusenapaul@gmail.com

Abstract

Power line inspection serves the purpose of fault finding and regular maintenance strategy, which are critical functions from an operational perspective. Modern computational intelligence, decreased cost in processing and communication components provides the opportunity to automate these inspections with the aim of adopting a predictive maintenance strategy. This paper deals with the design of a Power Line Tracking Unmanned Arial Vehicle. The literature review focuses on the navigation of the PLTUAV and selects the GPS, Gyroscope, and Accelerometer with correction mechanism done through Kalman filtering. The details of the overall system design and each of the subsystems are presented. Preliminary test results and analysis of the subsystems are also given.

Keywords
Power Line Inspection, Predictive Maintenance, Power Line Tracking Unmanned Arial Vehicle (PLTUAV).

1. INTRODUCTION

One of the biggest challenges that we, in South Africa, are currently facing is the interrupted supply of power. Theft of transmission and distribution cables is among the causes of these interruptions. This becomes a cause for concern to staff responsible for maintaining these lines. When the cable is stolen, the initial alert in control room arises from tripping of the circuit breaker. A very large portion of the line is taken out of service during this fault and it is difficult to localize the point at which the problem has occurred. Customers supplied by this part of the network are directly affected and do not have electricity. Field service technicians are dispatched to the area for fault finding. This is typically carried through visual inspection and technicians traverse the line to detect problems. Furthermore, such inspections become more challenging during adverse conditions such as inclement weather and/or poor visibility. This process of inspection and fault localization can take several hours to complete. Once the fault is localized, then only can the repair/maintenance work commence. The entire process can take several hours or days depending on the time taken to find and fix the fault.

The proposed Power Line Tracking Unmanned Arial Vehicle (PLTUAV) attempts to minimize the amount of time taken to search for the problem. This vehicle is in the form of a drone, does not require any pilot, tracks the power line and carries out real-time reporting to the Central Control System (CCS).

The proposed PLTUAV:

- First locates the Power Line with confirmation from the CCS
- Tracks the Power line to the direction specified by the user, in reference to the point of start

- Maintains the same distance from the Power Line during operation
- Avoid obstacles during flight
- Report the location to the CCS
- Relays the information of the line to the CCS in real-time which includes transmission of pictures and video.
- After tracking is complete (or recall), the PLTUAV returns to the CCS.

The research attempts to compare and find the best method to use for the self-navigation of the PLTUAV. It is here were we choose the Global Positioning System (GPS) with aid of Gyroscope and Accelerometer to do navigation of the PLTUAV.
The PLTUAV with the aforementioned capabilities has been built using two microcontrollers –i.e. the PIC32MX795F512L which controls the sensors, processing of data and sending out via the WiFi Module, a PIC18F45k22, which controls the drone motors, the gyroscope and monitors the battery. PLTUAV uses a GPS module to provide its location and movement data at all times. It uses distance sensors to detect obstacles and to land upon completion of the inspection. A camera module is also used to take the image of the line at all times. The gyroscope is used to for performing flight control and stabilization of the PLTUAV.

2. LITERATURE REVIEW

Navigation System for UAV

Zhuoning Dong, Wenbin Li, and Yanxing Zhou published a paper on autonomous navigation scheme for UAV in approach phase. The navigation information is obtained via integrated navigation systems. They use a Kalman filter to do a feedback error correction into their integrated navigation system. The components used on to build the integrated system are the strapdown inertial navigation system (SINS), electronic compass, optical flow, altimeter system and laser range finder. The simulations for this method, were found to run smoothly and give out good results interms of attiude angle, height and velocity. This design method was made for low attitude flying phase (Zhuoning Dong, 2016).

Sheng Bao, Jizhou Lai, Zang Chen, Pin Lyu, and Wenjing Chen looked at the Aerodynamic Model/INS/GPS failure-tolerant navigation method for multirotor UAVs based on federated Kalman filter(FKF). Their aerodynamic model has the capability to estimate the velocity of the multi-rotor UAV whilst the FKF fault are distinguished by the chi-square test. Now during the GPS failure, the sub-filter including the GPS is isolated, and when the aerodynamics has faults, the sub-filter including the aerodynamics model is isolated. This method improves the navigation accuracy and reliability of the multirotor and even cators for fault (Sheng Bao, 2017).

Zhang Liang, Lai Ji-Zhou, Shi Peng, Bao Sheng, and Liu Jian-Ye came with an improved MCS/INS integrated navigation algorithm for multi-rotor UAV in indoor flight. This system is designed to reduce the delays caused by motion capture system which provides high precision navigation at the cost of being slow. This method fuses the onboard inertial sensor information with the delayed measurement data to compensate for the error delay using Kalman filter. The experiment done on this paper proves to be success (Zhang Liang, 2016).

For navigation system, Zhuoning Dong and his friends came with paper that I could relate to with the outcome I require for my system. They use SINS, with Compass, Altimeter and laser range finder, these sensors are combined and corrected through Kalman filtering and were found to be working perferectly. Sheng Bao ‘s paper showed a very good techniques that ensure you never get to a point of not navigating due to errors on the GPS and their aerodynamics model, I did however find that the problem they are trying to solve on this paper will rarely affect me. What causes error data from GPS module is 1st, its antenna not having clear view sky and so not connecting to satellites, 2nd, it being to close to magnetic equipment, and 3rd, the actual GPS device being physically faulty, on the 1st one, power line are generally higher than the trees, the is even a clearance distance of above a meter to the trees and are outside, this drone is to be designed to hover above these wires and so the sky clearance would almost never be impeded. The 2nd point is the GPS unit is too be placed far away from magnetic motors, and 3rd, before every flight the GPS is to be tested properly for its functionality. These then makes Sheng Bao’s method an “over-kill” for my design. Zang Liang’s paper is indoors based design, but the method for combining the sensor data and do error correction is to be used in my paper.

The Methodology I am to use is to be similar to that of the first paper presented on this literature review, I am how going to use GPS instead of SINS, Gyroscope and Accelerometer instead of compass and range finder. The data given out will still be similar. To combine and correct the data I am to use Kalman filtering just like in the three papers

3. RESEARCH METHODOLOGY

Figure 1 shows the functional hardware layout of the Power line tracking System.

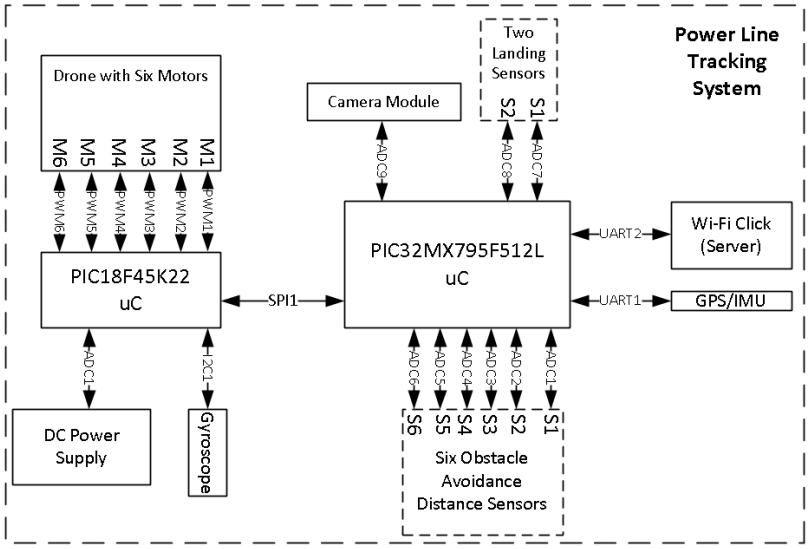

Figure 1: Hardware layout of the Power Line Tracking System

Error! Reference source not found. shows that the Power Line tracking system uses two microcontrollers, a PIC32MX795512L and PIC18F45K22, to control and monitor the airborne system. The components that make this control system are the Global Positioning System (GPS), which provides among other things the GPS coordinates of the platform at all times, the WiFi Plus Click Module, which sends all the data to the Central Control System via a built in antenna over a distance of 400 m, six distance sensors, which are used to detect obstacles around the platform, this way assisting in avoiding them, a gyroscope, which provides the orientation of the platform at all times, a UAV/Drone made up of 6 motors, that is used as our flying platform, and two distance sensors that are facing the ground, that are used to assist the drone in landing and maintaining a certain distance above ground. The system is also interfaced with our DC power supply, were it monitors the battery capacity at all times. The subcomponents of the Power line tracking system have been thoroughly explained and results shared.

3.1. Microcontroller

Two microcontrollers are used for this control System. The PIC18F45K22, which controls the drone and the gyroscope, and the PIC32MX795F512L, which controls all the other sensors and the downlink communication. The microntrollers are the central controllers of the ECM Control System. All the information from the different sensors, PIC18F45K22 and Systems are collected, processed, packaged and sent over the WiFi link using the PIC32MX795F512L microcontroller. **Error! Reference source not found.** shows both the PIC32MX795F512L and PIC18F45K22 microcontrollers. All embedded software developed for these micros was performed using MikroC PRO.

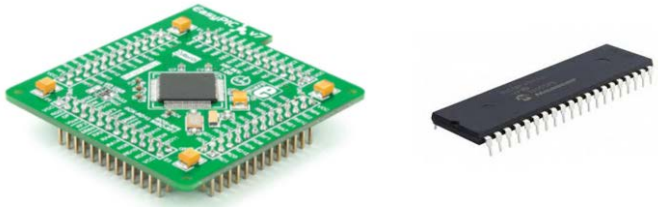

Figure 2: PIC32MX795F512L (MikroElektronika, 100-pin TQFP PT ETHERNET with PIC32MX795F512L) (Left) and PIC18F45K22 (Right) MCU from MikroElektronika

3.2. Global Positioning Unit

In order to determine the Drone's position, speed and heading in space a LIA 6 GPS Module, packaged as a GPS Click from MikroElektronika is used. The Drone uses the information from this module to continuously calculate its positon relative to the CCS in order to accurately point its own antenna towards the CCS when transferring data downstream. The data from this GPS module is key in assisting the Drone to know where it is at all times relative to the CCS, this allows it to even "come home" after the flight. The module is shown in **Error! Reference source not found.**.

Figure 3: GPS Click from MikroElektronika (MikroElectronika, GPS Click, 2018)

This module is powered by 3.3V. On power-up this module operates at a Board rate of 9600, it gives out lots of information at a frequency of 5Hz. On that data, you get all of the data that one requires. In order to maximize the accuracy reading of the position of the platform, one needed to ensure that the GPS update rate was very high, one felt 5Hz wasn't fast enough, also after intensive testing, one found that the data kept on streaming in from the GPS to the Microcontroller was in text format, this meant that one needed some kind of processing inside the Microcontroller before one could use this data, this then automatically meant I was decreasing the update rate. The mode at which this GPS was operating in was the National Marine Electronics Association (NMEA) mode that operated at a maximum frequency of 5Hz. This GPS had another mode, called the U-Blox, which operated at a Maximum rate of 13Hz, and the data it gave out was in ASCII form. The mode was the most desirable and so it was chosen for this project. This then compelled one to deactivate the NMEA messages and Poll for the data we require at 13Hz frequency. From the GPS Click module, the following information has been extracted:

- GPS Longitude
- GPS Latitude
- GPS Altitude
- Number of Satellites the GPS is connected to
- GPS Heading
- GPS Speed in the X,Y and Z Direction

The GPS Longitude (Long), Latitude (Lat), and Altitude (Alt) gave us the geographical location of our airborne platform at all times. After intensive testing, it was found that the accuracy of the Long, Lat, and Alt was heavily depended on the number of satellites the GPS module was connected to, hence we decided to constantly check the number of satellites the GPS was locked to that way we know our Long, Lat and Alt are accurate. We also used the

GPS heading to keep check at all times were the drone is facing relative to the CCS, this became a great assistance in pointing the antenna it the direction of the CCS when transmitting data via telemetry radio. The North East Down (NED) velocity and the Earth Centred Earth Fixed (ECEF) Velocities are read from the GPS unit and are compared to each other to get the most accurate solution. This information best informs us on the location of the drone and the speed at which it is moving.

This information was stored in four different registers, which one needed to poll the GPS to get them. This registers are the following:

- **NAV-POSLLH -** This is called the Geodetic Position Solution, this register, after being polled, come with structure of 36 bytes, in that 36 bytes, 12 of those bytes represent the Long, Lat, and Alt (4 bytes each)
- **RXM-SVSI –** This is called the SV Status Information. This register, after being polled gives out 440 bytes of data, from that one only uses a single byte that tells us the number of satellites the GPS unit is locked too.
- **NAV-VELNED-** This register is called the velocity solution in NED form. After being polled, this register gives out a structure with 44 bytes, on these 44, 16 bytes. 4 bytes are for the velocity in the North Direction, 4 byte for Velocity in the East Direction, 4 bytes for the velocity on the Down direction, and the last 4 bytes are for the heading.
- **NAV-VELECEF –** This register gives the velocity solution in ECEF. After being polled, this register gives out a structure of 28 bytes. On the 28 bytes, 12 bytes are used for the velocity in the X, Y, and Z direction, all carrying 4 bytes each.

Now during start, on the code we first disable the NMEA messages. The piece of code below shows the instruction sent from the microntroller to the GPS to disable the NMEA messages.

```
DIsable NMEA GGA Messages
delay 50ms
DIsable NMEA GLL Messages
delay 50ms
DIsable NMEA GSA Messages
delay 50ms
DIsable NMEA GSV Messages
delay 50ms
DIsable NMEA RMC Messages
delay 50ms
DIsable NMEA VTG Messages
delay 50ms
```

After these messages are sent, sends instruction that increase the update rate of the GPS module to 76ms (13Hz) and also increase the baud rate of the GPS module from 9600 to 230400, one also ensures that the microcontroller is also operating at the same baud rate. This increases the speed of the GPS module. Below shows the piece of code that does this.

```
Set GPS Baud rate to 230400
delay 50ms
Change uC UART5 Baud rate to 230400
delay 50ms
Change uC UART2 Baud rate to 230400
delay 50ms
Set GPS Update rate to 76ms (13 Hz)
delay 50ms
```

From this, the polling messages are sent, one polling message and the GPS responds almost immediately with the structures of data serially via UART. Immediately when the data is gets to the micro, the micro reads the data and stores it to a register allocated to that specific message. Piece of code below shows the polling of the POSLLH register, and code to receive and store the data received from the GPS

```
Poll POSLLH
Wait until UART2 Data is ready
Read UART2 data to 1st byte of POSLLH register
if(1st byte of POSLLH register = 0xB5)
{
    if(UART2 Data is Ready)
    {
        Read UART2 data to 2nd byte of POSLLH register
        if(2nd byte of POSLLH register = 0x62)
        {
            if(UART2 Data is Ready)
            {
                Read UART2 data to 3rd byte of POSLLH register
                if(3rd byte of POSLLH register = 0x01)
                {
                    if(UART2 Data is Ready)
                    {
                        Read UART2 data to 4th byte of POSLLH register
                        if(4th byte of POSLLH register = 0x02)
                        {
                            for(i = 4, i <36, increment i by 1)
                            {
                                if(UART2 Data is Ready)
                                Read UART2 data to ith byte of POSLLH register
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Error! Reference source not found. shows the four polling messages, represented by the yellow channel, and the GPS responses, shown on the Blue channel. **Error! Reference source not found.** below is an image of the terminal on the screen, showing the outputs from the GPS, this is to just show that the GPS is working perfectly and is giving out through readings.

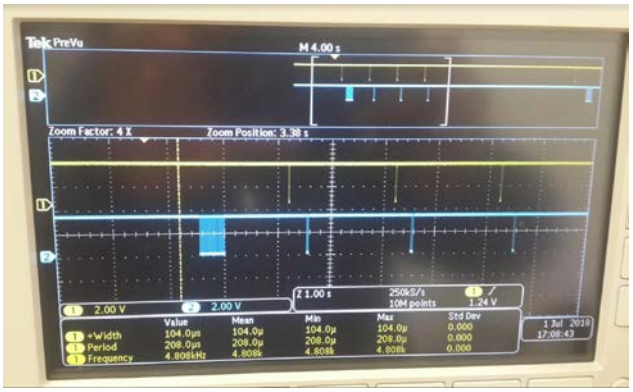

Figure 4: Four poling Messages (Yellow Channel) to the GPS and the responses (Blue channel) from the GPS

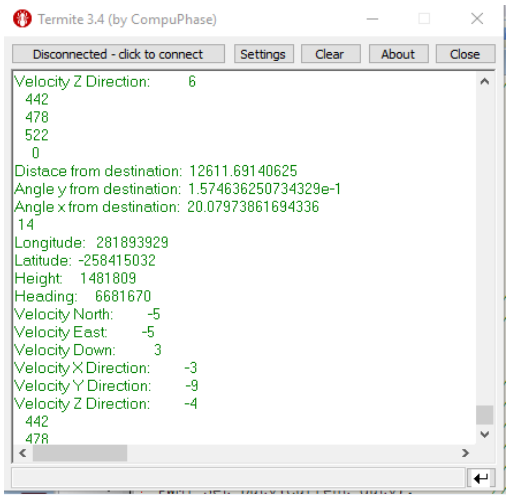

Figure 5: GPS Outputs shown on termite, on the PC Monitor

3.3. **Distance Sensors**

The drone is designed to be autonomous, because of this it is designed with the capability of detecting obstacles during flight and be able to avoid them, it should have the capability of taking off from the ground and landing with minimal assistance from a human being. To achieve Obstacle avoidance, the drone needs to first detect if there is an obstacle coming towards it, same for the landing, the drone needs to know its distance from the ground in order to adjust its speed accordingly, without landing to rough and damaging the equipment on board. Distance sensors are used to assist with this.

A SHARP optoelectronic distance sensor, named GP2Y0A02YK, is used for collision avoidance of the sensors. This sensor powered by 5 Volts(V), its general current consumption is 33mA. This operates at a range of 20 to 150 cm. the response time for this sensor is 39 milliseconds (ms). **Error! Reference source not found.** shows the GP2Y0A02YK distance sensor.

Figure 6: GP2Y0A02YK distance sensor (RobotBits, 2018)

Six of these distance sensors are used, these are attached underneath each propeller of the drone. They are facing outwards to detect anything that is around the drone. This type of sensor gives out an analogue output that is from 2.5 V, that is when the objected to be detected is at 20 cm, and gives just less than 0.5 V for a distance that is at 150 cm. The functionality curve of this sensor is shown in **Error! Reference source not found.**.

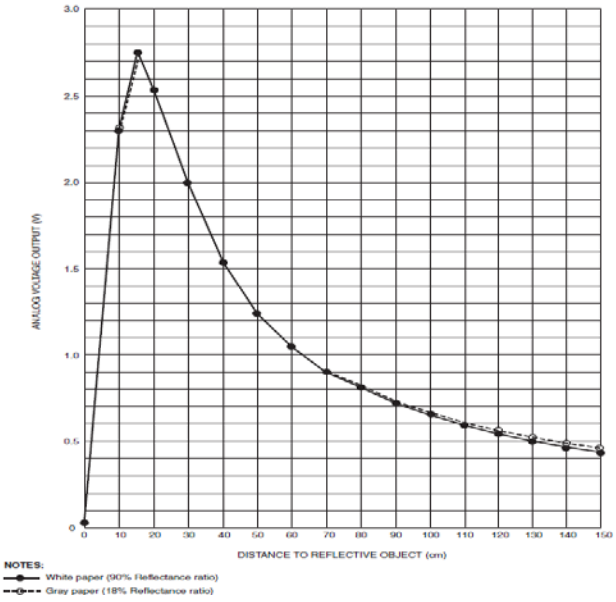

Figure 7: Functional output curve of GP2Y0A02YK (SHARP, 2018)

For landing purposes, two distance sensors, also from SHARP, with the name GP2Y0A710K0F, are used. These have a range of 100 to 550 cm. these sensors are also powered by 5 V. Their current consumption when operational is 30mA. This sensor gives out an analogue output. At an object that is 100 cm away, this sensor gives out a voltage of 2.5 V when sensing that object, at a distance of 550 cm, this sensor gives out a voltage that is 1.4 V. The Output characteristics of this sensor is shown in **Error! Reference source not found.**.

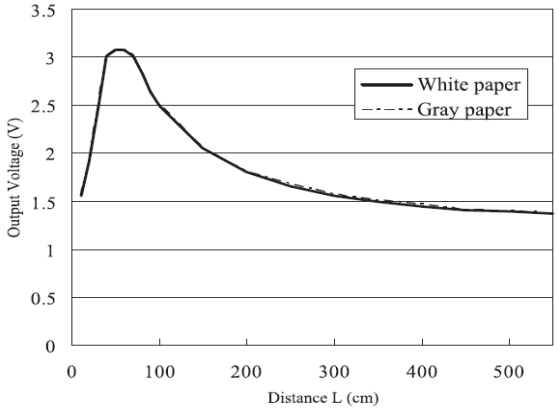

Figure 8: GP2Y0A710K0F Output Characteristics (SHARP, GP2Y0A710K0F)

From **Error! Reference source not found.** and **Error! Reference source not found.**it is clear that the output of these sensors operates between 0 and 3.3 V. to read this data from the microcontroller, internal Analogue to Digital Converter (ADC) of the PIC32MX795F512L microcontroller is used. This micro consists of 10 bit ADC and can handle up to 16 analogue inputs. The reference voltage used for these sensors was a minimum of 0 V to a maximum of 3.3V. This means that if the sensor gives an analogue voltage output of 0V the Digital value from the micro would be 0, and if the sensor gives out 3.3 V output, the digital value in the would be 1023. The line of code to initialize the one ADC, with up to 16 input channels is shown below.

```
Initialize ADC
delay 50ms
```

This ADC is given a 100ms delay to initialize properly. The reading of the ADC is reflected on the lines of code below.

```
Read ADC Connected to Sensor 1
Read ADC Connected to Sensor 2
Read ADC Connected to Sensor 3
Read ADC Connected to Sensor 4
Read ADC Connected to Sensor 5
Read ADC Connected to Sensor 6
Read ADC Connected to Sensor 7
Read ADC Connected to Sensor 8
```

The variables adc_sensor_1 to 8 store the digital value that represents the output voltage of the sensor. **Error! Reference source not found.** shows the outputs of one obstacle avoidance sensor; this is the output on the left, and one landing sensor, shown on the right. The obstacle sensor was tested by putting a book in front of it at a relatively close distance hence the change in values of the output from a value of 5 to 869. The landing sensor was pointed to wall a bit further and it gave these values.

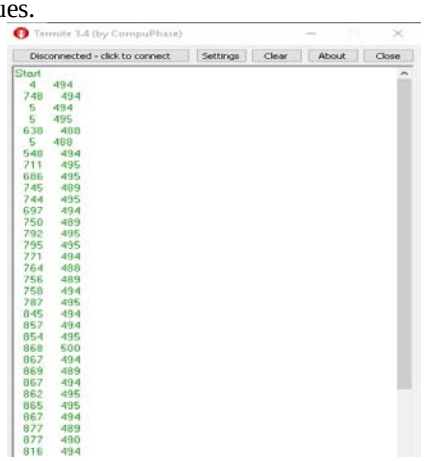

Figure 9: Output of 1 Obstacle Sensor (Left) and 1 Landing Sensor (Right)

3.4. **WIFI Plus Click**

The Wi-Fi Plus Click was chosen because of its small size and the fact that it has a built in PCB antenna. The Wi-Fi click features the MRF24WBOMA which operates at 2.4Ghz frequency and adheres to IEEE 802.11 std. It also has MCW1001 companion controller with TCP/IP stack and 802.11 connection manager. The Wi-Fi PLUS click is powered by 3.3V, it communicates with Microcontroller Unit (MCU) over a UART communication protocol. The MRF24WBOMA RF transceiver module contains integrated PCB antenna with range up too 400m. the typical power consumption, during operation, of this is 250uA.

Figure 10: WiFi PLUS Click from MikroElektronika (MikroElektronika, 2018)

The WiFi module was set to be a slave on the radar side. The code below shows the setting up of the variables to be used to set up this network, the naming of network Number of channels, the SSID of the network, the Mac address of the slave WiFi module, the IP address, Gateway address, and the IP Mask. Also shown in this piece if code is the home's Mac and IP address, home in this case being the CCS.

```
Set variable channels to 11 bytes = (1,2,3,4,5,6,7,8,9,10,11)
Set variable strSSID to 11 bytes = (Drone)
set variable myMacAddr to 6 bytes = (0x1A,0x1B,0x1C,0x1D,0x1E,0x1F)
set variable myIpAddr to 4 bytes = (146,64,247,8)
set variable gwIpAddr to 4 bytes = (0,0,0,0)
Set variable IpMask to 4 bytes = (255,255,255,0)

set variable HomeMAcAddr to 6 bytes = (0x11,0x12,0x13,0x14,0x15,0x16)
set variable HomeIpAddr to 4 bytes = (146,64,247,10)

set variable remoteIpAddr to 4 bytes
```

The piece of code below initialize the Wi-Fi module, it first give the module a 20 seconds delay to initialize properly, it then sets the mode of operation of the WiFi module to Infrastructure, this will ensure that the code has a client and a server during operation. When this has been done successfully, it flickers a Light Emitting Diode (LED). After this it goes and sets the channel list, sets the security of communication to open, meaning the is no security, setting the SSID of network. After this the code sets all the specific addresses of this network. The code then goes and set the socket allocation, in this it informs the system that the will be one client and server on the network, the server will receive and transmit a maximum of 2048 bytes, this effectively tells the WiFi module that it will be the server in this network.

```
Drone_InitWifi
{
    Put time to wait for Wifi module to 20s
    response =1

    while(response is not equal to 0)
    {
        response = Set mode to Infrastructure  //This will set response to 0 if this line has no error
    }

    Set Net_Wireless_MCW1001_SetNetworkMask to IpMask
    Set Net_Wireless_MCW1001_SetGatewayIP to gwIpAddr
    Set Net_Wireless_MCW1001_SetMAC to myMacAddr
    Set Net_Wireless_MCW1001_SetIP to myIpAddr
    Set Net_Wireless_MCW1001_SetArpTime to 1
    Set Net_Wireless_MCW1001_SetRetryCount to 5
    Set Net_Wireless_MCW1001_SocketAllocate to 1 server, 1 client, 2048 server receive bytes, 2048 server transmit bytes

}
```

The code below goes and scans the number of networks that are seen by this WiFi module. Response in the first line of code come back with the number of the networks seen by the WiFi module, if response is 3 for example, that means the WiFi module sees three wireless networks. A loop Is then created that will go through all the networks seen by the Wi-Fi module. The that say **Net_Wireless_MCW1001_GetScanResult(i)** gets the full data of the **i**th network, it comes back with the SSID of that network, the IP address of that network, the length of the SSID of that network . After it has received these, it copies the SSID of that network to a register called **strTmp**.

The code then compares the SSID of the scanned network with its own SSID; obviously, it is looking for the wireless network that shares its own SSID as this will be the master WiFi module from the CCS.

```
Scan Available Wireless Networks  //This will return with the Number of Wireless Networks seen

for(i = 1, i < Number_of_Wireless_Networks , increment i by 1)
{
    response = Net_Wireless_MCW_GetScanResults(i)
    Strncpy(strTmp, Net_Wireless_MCW1001_ScanResult.SSID, Net_Wireless_MCW1001_ScanResult.SSIDLength) //This copies SSID to Variable strTmp
    if(strncmp(Net_Wireless_MCW1001_ScanResult.SSID, strSSID, Net_Wireless_MCW1001_ScanResult.SSIDLength) == 0)
    {
        We got the write Network
        break
    }


}
```

After finding the right SSID, the WiFi module connects with CCS WiFi Module, after this it creates a socket for the communication, it the n binds the socket, what this effectively does is it associates its IP address with this socket, meaning when data is sent to the IP address of the module, it comes through this socket.

```
Drone_ConnectToWifi
{
    WiFi Module to wait 1s
    Net_Wireless_MCW1001_Connect
    Net_Wireless_MCW1001_Properties.networkStatus = _NET_WIRELESS_MCW1001_STATUS_NOTCONN_STATIC_IP;
    //Net_Wireless_MCW1001_Properties.macAddress = homeMacAddr;
    //Net_Wireless_MCW1001_Properties.ipAddress = homeIpAddr;
    //Net_Wireless_MCW1001_Properties.networkMask = ipMask;
    //Net_Wireless_MCW1001_Properties.gatewayAddress = gwIpAddr;

    Infinity loop
        {
            Net_Wireless_MCW1001_GetNetworkStatus();
            if(NetworkStatus is connected to Static IP)
                break
        }
    //Create all necessary WiFi Network Sockets
    socketHandle = 0;
    localPort = 6000;
    localPort = 6000;
    Net_Wireless_MCW1001_TimeToWait = 1;
    while(Net_Wireless_MCW1001_SocketCreate(&socketHandle, _NET_WIRELESS_MCW1001_SOCKET_TYPE_UDP));
    while(Net_Wireless_MCW1001_SocketBind(socketHandle, &localPort, &bindResponse));
    while(Net_Wireless_MCW1001_SetArpTime(0));

}
```

From this point, the WiFi module is ready to read data

```
void Drone_WifiReadPacket(void)
{
  response=Net_Wireless_MCW1001_UDP_ReadBytes(&socketHandle, RXPKTSIZEMAX, &remoteport, remoteIpAdd, s_home_packet.a_home, &numOfReceiveBytes);
}
```

3.5. Unmanned Arial Vehicle

The frame used for this drone is called the S550 Multi-Rotor Air Frame. ***Error! Reference source not found.*** shows the frame. The Motors used are the SURPASS high performance brushless electric motors, ***Error! Reference source***

not found. shows the motor. A total of six of these motors are required by this structure and so are used, commercially available propellers were used. To control the motors, hobbywing x-rotor 20a motor controller is used, this shown in ***Error! Reference source not found.***. This Controller is powered by 15V in our circuitry, it switches the motor using only Pulse Width Modulation (PWM). The frequency in which this controller operates on is 500 Hz, the motors run when the duty cycle of the pulse is between 1000 to 2000 ms. The physical connection between the motor, Motor Controller, and the Microcontroller is shown in

Figure 11: S550 Multi-Rotor Air Frame (Water, 2018)

Figure 12: SURPASS High Performance brushless electric motor

Figure 13: Hobbywing x-rotor 20A Motor Controller (Technology, 2018)

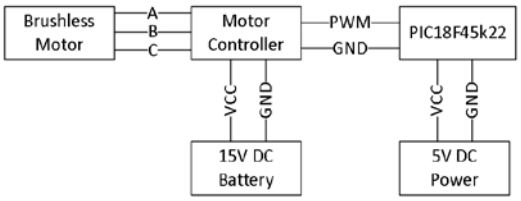

Figure 14: Connection from Microcontroller to Brushless Motor

The following code shows the increment of the duty cycle for each of the PWM and this effectively increases the speed of the motors.

```
while(1)
{
  Delay_ms(1000);
  current_duty1++;
  PWM1_Set_Duty(current_duty1);
  PWM2_Set_Duty(current_duty1);
  PWM3_Set_Duty(current_duty1);
  PWM4_Set_Duty(current_duty1);
  PWM5_Set_Duty(current_duty1);

}
```

Error! Reference source not found. shows the PWM signal that drives the motors of the Drone.

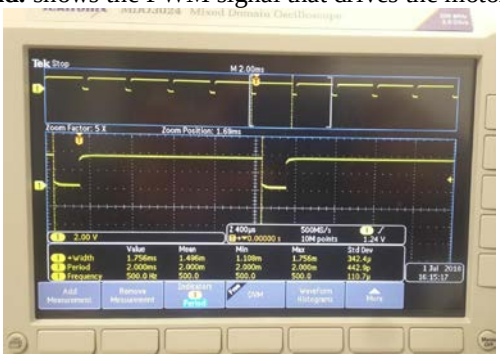

Figure 15: PWM Signal that drives the PLTUAV Motors

Figure 16: Preliminary testing of PLTUAV

4. Conclusion

Almost all the components of the drone, individually, were tested and were found to be function as per the expectation. The only component not tested was the camera module, this will be thoroughly explained on the next paper to come. This paper shows how these components were tested and shows the results for the components that can be proven that they work. The future work to be done will focus more on the flight control of the Drone and the Image processing from the camera module and the integration and results of the full Power line Tracking UAV.

5. References

1. MikroElectronika. (n.d.). *100 -pin TQFP PT ETHERNET with PIC32MX795F512L*. Retrieved from https://eu.mouser.com/Semiconductors/Engineering-Development-Tools/Embedded-Development-Tools/Datasheets/_/N-cxwd2?keyword=PIC32MX795F512L
2. MikroElectronika. (n.d.). *100-pin TQFP PT ETHERNET with PIC32MX795F512L.* Germany: Mouser Electronics.
3. MikroElectronika. (2018, July 4). *GPS Click*. Retrieved from GPS Click: https://www.mikroe.com/gps-click

4. MikroElektronika. (2018, June 10). *WiFi Plus Click*. Retrieved from https://www.mikroe.com/wifi-plus-click
5. RobotBits. (2018, June 25). *RObot Bits*. Retrieved from RobotBits: https://robotbits.com/20-150cm-ir-range-finder-gp2y0a02yk.html
6. SHARP. (2018). *GP2Y0A02YK Optoelectronics Device*. Japan.
7. SHARP. (n.d.). *GP2Y0A710K0F*. Japan: SHARP.
8. Sheng Bao, J. L. (2017). Aerodynamic model/INS/GPS failure-tolarant Navigation Method for multirotor UAVs based on Federated kalman filter. (pp. 1121-1125). Nanjing, China: College of Automation Engineering .
9. Technology, S. F. (2018, June 2). *AliExpress*. Retrieved from https://www.aliexpress.com/item/F17723-Hobbywing-XRotor-Lipo-3-4S-Long-wire-20A-Brushless-ESC-No-BEC-high-refresh-rate/32664708916.html
10. Water, G. (2018, June 28). *s550 HEXCOPTER FRAM KIT*. Retrieved from http://offthegridsun.com/index.php?main_page=product_info&cPath=5&products_id=807&zenid=8d775d0b1beca3e853b1926f5765e795
11. Zhang Liang, L. J.-Z.-Y. (2016). An improved MCS/INS integrated navigation algorithm for multi-rotor UAV in indoor flight. *Navigation and Control Conference* (pp. 12-14). Nanjing, China: Proceedings of 2016 IEEE Chinese Guidance.
12. Zhuoning Dong, W. L. (2016). An autonomous navigation scheme for UAV in approach phase. *Navigation and Control Conference* (pp. 12-14). Nanjing, China: Proceedings of 2016 IEEE Chinese Guidance.

6. Biographies

Mcebisi Solilo is a full time Masters student at University of Johannesburg, at the department of Electrical and Electronic Engineering Technology. He works as a Senior Engineer for a defence based company called Sovereignty Systems that specializes in Electronic Counter Measure Systems and Radar signal generation. He earned his BSc in Electrical Engineering from University of KwaZulu Natal in the School of Electrical, Electronic and Computer Engineering. His research Interest is on how Unmanned Arial Vehicles could improve the society.

Wesley Doorsamy received the B.Sc, M.Sc. and Ph.d. Degrees in Electrical Engineering, and a Postgraduate Diploma in Higher Education from the University of the Witwatersrand in Johannesburg, South Africa in 2008, 2013, 2015 and 2017, respectively. He is currently a senior lecturer at the University of Johannesburg and is an active member of both the Institute of Electrical and Electronics Engineers (IEEE) and South African Institute of Electrical Engineers. In 2017, he was awarded the affiliate membership with the African Academy of Sciences (AAS). His research interests are signal processing, machine learning and pattern recognition, applied probability and statistics.

Babu Sena Paul received his B.Tech and M.Tech degree in Radio physics and Electronics from the University of Calcutta, India. He was with Philips India Ltd from 1999-2000. He received his Ph.D. degree from the Department of Electronics and Communication Engineering, Indian Institute of Technology. He has attended and published over sixty research papers in international and national conferences, symposiums and peer reviewed journals. He has successfully supervised several postgraduate students and post-doctoral research fellows. He joined the University of Johannesburg in 2010. He has served as the Head of the Department at the Department of Electrical and Electronic Engineering Technology, University of Johannesburg from 2015 to March 2018. He is the currently serving as the Director to the Institute for Intelligent Systems, University of Johannesburg.