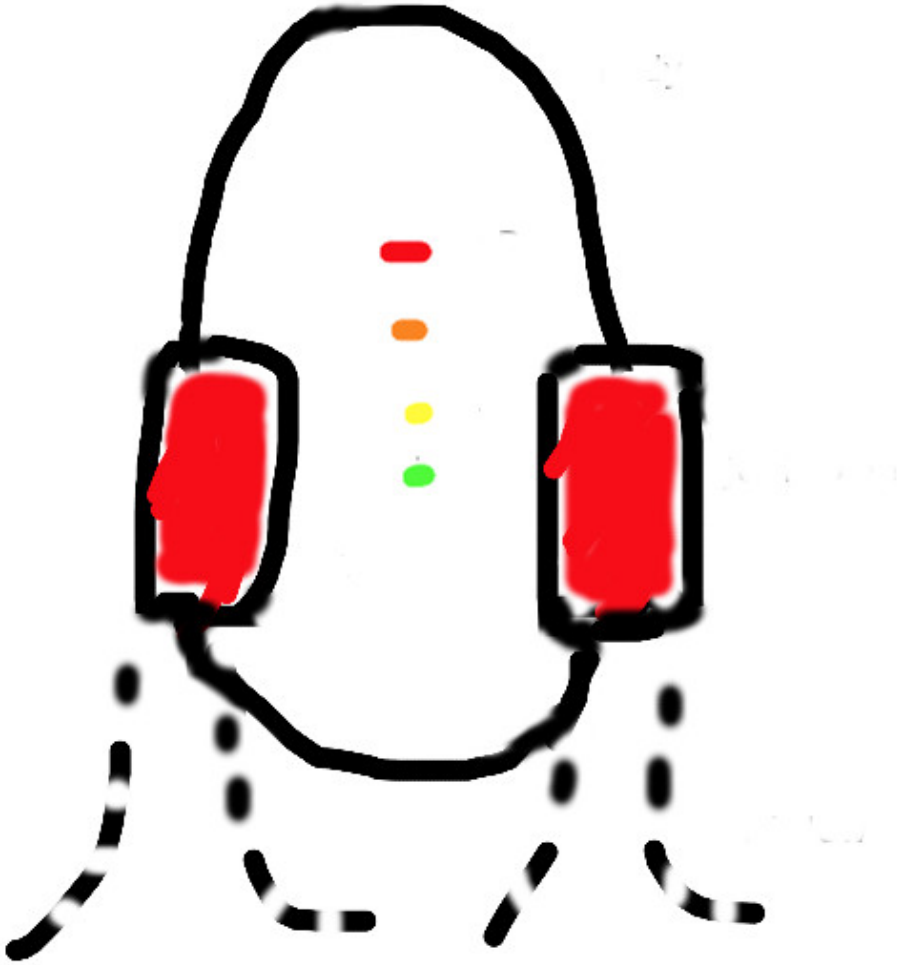
Artificial Intelligence

Swarm of Drones

X. Wang

Open Project



TU Delft Delft University of Technology

Delft University of Technology

Artificial Intelligence
Swarm of Drones

Open Project

X. Wang

October 29, 2017

Applied Sciences · Delft University of Technology

Abstract

Artificial Intelligence (AI) is an interesting topic nowadays. But how can we incorporate an AI software with a hardware? To answer this question we propose two drone projects which combine an AI software into a drone. We would like to cover both the software and the hardware of those drones in this report. Our drones are able to charge themselves from a charging station. At the later phase we would like to explain how to build a swarm of drones which communicates with each other.

Table of Contents

List of Figures

Preface

This document is written for you as a guide to better understand about our project. Our project is about Artificial Intelligence (AI) and drones. This project is carried out by three of us. We are two physics and one designer. We are motivated to run this project, because it helps us to gain more knowledge inside this field. The knowledge about how AI works and how to build a drone. This project gives me in person a good experience to work in a team, because I was always struggling to do that. I would like to thank my teammates Eng. J. Asmus and M.H.A. van Daalen for their support for this project. Eng. J. Asmus has a master degree in Applied Physics, and M.H.A. van Daalen owns a 3D design company. We use our spare time to discus about our project and solve our problems together. Especially M.H.A. van Daalen have made all the 3D designs.

X. Wang

October 29, 2017

Chapter 1

Introduction

Drone is a complex flying object which uses motors and propellers to stay steady in the air. The speed of the motors are controllered by the ESC's (electronic speed controller). PID controller is a simplest controller that can stablize the position of the drone in the air. PID stands for propertional, integral and derivative. It uses sensors to calculate the rotation and position of the drone in the air, and sends the commands to the ESC's to adjust the speed of the motors for balancing. When we have moving parts inside a drone, then we have to deal with the mechanical noise of the moving parts. Kalman Filter helps us to gain more accurate measurement of the rotation and position of the drone from the multiple sensors.

First we figure out how to build a drone that acts like a human. Then we would like to build a swarm of drones that forms an entity. That entity will carry out tasks which are more complex. Artifical Intelligence (AI) is a man made intelligence which makes computer acts like a human. It includes machine learning. That will increase the knowledge of the drone over the time. In orde to keep the drone functioning. We need a self charging station where the drone charges by itself to provide the power.

In order to achieve that we have to create an AI program that controls the drone, and lets it learns from the environment by using sensors (e.g. camera, microphone etc). We are using statistice approach to simulate the intelligence. The drone will be able to use human language to communicate with us. It imitates a human for the interactions. After that we would like to use the swarm technology to build a swarm of drones which interact with each other to do something together.

We first start to talk about how to apply Artifiical Intelligence in our drones. Then we present two designs of a drone. But before we start we first explain in this chapter the PID controller, Kalman Filter and Self Charging Station in details.

1-1 PID controller

The general formula for the control variable of the PID controller in terms of $e(t)$ (error value) is:

$$u(t) = K_p e(t) + K_i \int_0^\tau e(\tau)\mathrm{d}\tau + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{1-1}$$

where K_p, K_i and K_d are the coefficients of the proportional, integral and derivative terms respectively.

In the discrate time series the PID controller can be represent by a PSD (propertional-summation-derivative) controller:

$$u[j] = K_p e[j] + K_i \sum_{n=0}^{j} e[n]\Delta t + K_d \frac{\mathrm{d}}{\mathrm{d}t} e[j] \tag{1-2}$$

where Δt is the time step in the discrate time series.

Since the setpoint of $e(t)$ is a constant therefore the derivative action can be written in term of the $PV(t)$ (process variable):

$$u[j] = K_p e[j] + K_i \sum_{n=0}^{j} e[n]\Delta t - K_d \frac{\mathrm{d}}{\mathrm{d}t} PV[j] \tag{1-3}$$

A C++ program for the PSD controller is given as:

```
/* Compute() **********************************************************
 *   Here is where the magic happens. This function calculates the
 *   output of each run.
 *********************************************************************/
void PID::Compute(double dt, float v)
{
   /*Compute all the working error variables*/
   double input = *myInput;
   double error = *mySetpoint - input;
   double percentage = 25
   STerm += (Ki * error) * dt;
   if(STerm > outMax*percentage/100) STerm = outMax*percentage/100;
   else if(STerm < outMin*percentage/100) STerm = outMin*percentage/100;
   double dInput = (input - lastInput);

   /*Compute PID Output*/
   double output = Kp * error + STerm - Kd * v;
   if(output > outMax) output = outMax;
   else if(output < outMin) output = outMin;
   *myOutput = output;

   /*Remember some variables for next time*/
   lastInput = input;
}
```

We have introduced two limits inside this program, because without those limits the program is unstable. The first limit is the maximum and minimum value of the output, then we also limit the summation term within 25% of the maximum and minimum value of the output. Because this term changes slowly and small respect to other terms. The most important part is to find the value of K_p, K_i and K_d. We tunned our drone by hands in order to find those parameters.

1-2 Kalman Filter

We have three sources of measurement: gyroscope, accelerameter and magnetrometer. From those three sources we would like to know the exact rotation and position of the drone. To combine those measurments we are using Kalman Filter to do that. From gyroscope we know the change rate of quantities which are measured from accelerameter and magnetrometer. From the accelerameter we know the gravitation vector, and from the magnetrometer we know the direction of north pole of the earth. The Arduino code which uses the Kalman Filter to measure the rotation and position is given as:

```
1  void loop() {
2    // #################################################################
3    // ---- readout sensors ------------------------------------------
4    // #################################################################
5    lsm303.read();
6    gyro.read();
7
8    // #################################################################
9    // ---- calculation ----------------------------------------------
10   // #################################################################
11   uint32_t start = micros();
12   double dt = (double)(start - timer) / 1000000; // Calculate delta time
         in seconds
13   timer = start;
14   N = (LSM303::vector<double>){lsm303.m.x-offset.x,lsm303.m.y-offset.y,
         lsm303.m.z-offset.z};
15   lm = sqrt(LSM303::vector_dot(&N,&N));
16   la = sqrt(LSM303::vector_dot(&lsm303.a,&lsm303.a));
17   if(la>0){
18     acc = (LSM303::vector<double>){
19       kalAcc.x.get(lsm303.a.x / la, (lsm303.a.y * gyro.data.z - lsm303.a.
             z * gyro.data.y) * DEG_TO_RAD / la, dt),
20       kalAcc.y.get(lsm303.a.y / la, (lsm303.a.z * gyro.data.x - lsm303.a.
             x * gyro.data.z) * DEG_TO_RAD / la, dt),
21       kalAcc.z.get(lsm303.a.z / la, (lsm303.a.x * gyro.data.y - lsm303.a.
             y * gyro.data.x) * DEG_TO_RAD / la, dt)
22     };
23     if(lm>0){
24       mag = (LSM303::vector<double>){
25         kalMag.x.get(N.x / lm, (N.y * gyro.data.z - N.z * gyro.data.y) *
               DEG_TO_RAD / lm, dt),
26         kalMag.y.get(N.y / lm, (N.z * gyro.data.x - N.x * gyro.data.z) *
               DEG_TO_RAD / lm, dt),
```

```
27          kalMag.z.get(N.z / lm, (N.x * gyro.data.y - N.y * gyro.data.x) *
                DEG_TO_RAD / lm, dt)
28        };
29        LSM303::vector_cross(&mag,&acc,&E);
30        theta = atan2(E.y,E.x)-angle;
31        theta = atan2(sin(theta),cos(theta));
32
33        le = sqrt(LSM303::vector_dot(&E,&E));
34        E = (LSM303::vector<double>){
35            E.x / le,
36            E.y / le,
37            E.z / le
38        };
39        LSM303::vector_cross(&acc,&E,&N);
40      }
41    }
42  }
```

1-3 Self Charging Station

We are using solar power to supply the power to the charging station. We charge the drone wirelessly from the station. We use two webcams to track the drone. By using this tracking system we let drone lands on the charging station. We use an Off Grid Solar Power System. For this system we need a solar panel, a charge controller and a battery. The battery is used to save the eletricity which is generated by solar panel. The battery and solar panel powered the whole self charging station and the charging for the drone.

Chapter 2

Artificial Intelligence

Fist we explain the theory about Artificial Intelligence (AI). Then we are going to explain the AI software in details. At the end we discuss the applications of AI software.

2-1 Theory of AI

AI is a concept that we build an intelligence into a machine. There are many different approach to build an AI e.g. brain simulations, soft computing, statical etc. There is still many question remain answered. The first calculating machine appears around 1623. The use of AI start in the late 1990s. By the mid 2010s, machine learning start booming all around the world.

The most important part about AI is the Machine Learning, because to hard program the knowledge into a machine is difficult. Therefore we have to find a way to let the machine gain the knowledge by itself. In order to let a machine to learn a human language we have to let the machine interacts with humans. We known that interaction is a best way to learn a language. To do so we simulate the human brain. We simulate the way how a brain functions. We are using the statistic approach to solve this problem. because we would like to introduce the randomness into the AI software, because we as a human is most of the time unpredictable.

We first store the information which comes from human into a memory. We then let the machine randomly outputs the information from the memory. To generate an output for an given input we have to process the messages by a logic unit. We are using induction and fuzzy logic to process those messages. because induction is a way which we use to reason. Fuzzy logic is a way to step out the binary thinking. Induction is a logic use all observation to prediction the future outcome. fuzzy logic uses the fuzziness of a quantity to make a decision. We use the stored information inside the memory to predict the output of an input. We use the fuzzy logic to generate a output for a known input. We teach the robot by interaction with the robot.

2-2 AI Software

We have written this software in Java (see figure 2-1). You can get the source code by using git with following command: git clone git://s55969da3.adsl.online.nl/java/AIRobot.git.

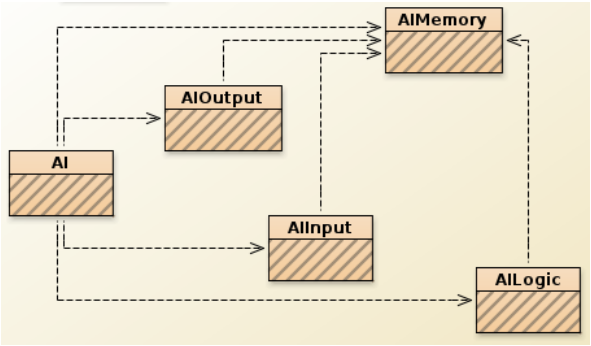


Figure 2-1: The diagram of the AI software.

There are four main classes inside this program: Output, Input, Memory and Logic Class. Output handles everything which is related to information that is present to the outside world. Input handles everything which is related to information that comes from the outside world. Memory is the place where the information is stored. Logic is the place where information is processed. AI class combines all classes together to function as a whole system.

2-3 Applications

We can build a chat and learning drone in this way. On later stage we would like to translate the sound into the text for processing, and then let machine to speak out the text from a speaker.

Chapter 3

Drone I

In this chapter we explain about the functions and the components of our first drone. In the first section we discus the design of the drone. Then we discus about the software of the drone. At the end we discus about the budget and planning of the drone.

3-1 Design of the Drone I

We have made a drawing of this drone (see figure 3-1).

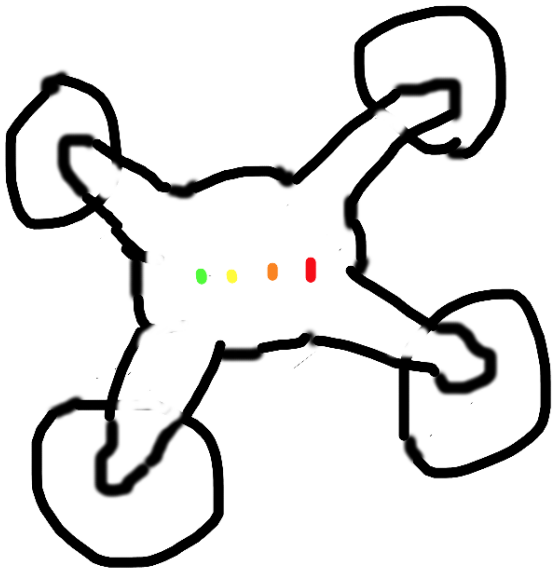


Figure 3-1: A drawing of drone I.

This drone has four blushless motors and four electronic speed controllers (ESC's). ESC's are controlled by an Arduino. We use an Ultrasonic Sensor to measure the height of the drone respect to the ground, and another four Ultrasonic Sensors to keep obstacle away from front, back, left and right side of our drone. We use gyroscope, accelerometer and magnetometer to stabilized our drone in the air. The battery has 4.0 Ah of capacity, therefore we expect about 5 to 10 minutes of fly time. For the future development we would like to use a self charging station to charge the drone.

3-2 Firmware

We use an Arduino to connect to our sensors and ESC's. We use a PID controller to stabilize our drone in the air. We use a low-pass filter to get out the high frequency noise. We use Kalman Filters to get more accurate measurements from our sensors. We have two parts in our Arduino code. The first part is the initialization of our software, and the second part is the processing loop. This loop takes 2-3 ms for each cycle. The Arduino setup code is given here below:

```
1  // -----------------------------------------------------------------
2
3  // #################################################################
4  // ---- initialisation ---------------------------------------------
5  // #################################################################
6
7  // +-- includes
8  #include <SFE_BMP180.h>
9  #include <Wire.h>
10 #include <LSM303.h>
11 #include <Adafruit_L3GD20.h>
12 #include <Kalman.h> // https://github.com/TKJElectronics/KalmanFilter
13 #include <PID_v1.h>
14 #include <Servo.h>
15 // --- includes
16
17 Servo myServo[4];
18
19 LSM303 lsm303;
20 SFE_BMP180 pressure;
21 Adafruit_L3GD20 gyro;
22
23 LSM303::vector<Kalman> kalAcc,kalMag;
24 LSM303::vector<double> E,N,mag;
25
26 LSM303::vector<double> setpoint,output,acc,Kp,Ki,Kd,offset,QAcc,QMag,
       QbiasAcc,QbiasMag,RAcc,RMag,Max,factor;
27 setpoint = (LSM303::vector<double>){   0   ,   0   ,   0   };
28 output   = (LSM303::vector<double>){   0   ,   0   ,   0   };
29 acc      = (LSM303::vector<double>){   0   ,   0   ,   1   };
30 Kp       = (LSM303::vector<double>){ 150   , 150   ,  70   };
31 Ki       = (LSM303::vector<double>){  30   ,  30   ,   6   };
32 Kd       = (LSM303::vector<double>){   1   ,   1   ,   2   };
```

```
33 offset   = (LSM303::vector<double>){ -651       , -827.5     , 272.5     };
34 QAcc     = (LSM303::vector<double>){    0.00001,    0.00001,    0.00001};
35 QMag     = (LSM303::vector<double>){    0.00001,    0.00001,    0.00001};
36 QbiasAcc = (LSM303::vector<double>){    0.0001 ,    0.0001 ,    0.0001 };
37 QbiasMag = (LSM303::vector<double>){    0.0001 ,    0.0001 ,    0.0001 };
38 RAcc     = (LSM303::vector<double>){    0.005  ,    0.005  ,    0.005  };
39 RMag     = (LSM303::vector<double>){    0.005  ,    0.005  ,    0.005  };
40 Max      = (LSM303::vector<double>){  200      ,  200      , 200      };
41 factor   = (LSM303::vector<double>){  900      ,  900      ,   0      };
42 LSM303::vector<int>    neutral  = (LSM303::vector<int>   ){ 1480      ,
       1500      ,   0      };
43
44 float la,lm,le;
45 int out = 0;
46 int startingPoint = 1400;
47 int timeout = 100000;           // in us
48 int wait = 300000;              // in us
49 double T = 20;                  // in degree Celsius
50 double P = 1013.2;              // in mbar
51 double theta = 0;               // in rad
52 const double phi = M_PI/4;      // in rad
53 double baseline = 1013.2;       // in mbar
54 double opticalX = 0;
55 double opticalY = 0;
56 double hight,angle;
57 uint32_t timer;
58 bool isSenderOn;
59 char statusPressure, statusTemperature;
60
61 PID myPIDx(&acc.x , &output.x, &setpoint.x, Kp.x, Ki.x, Kd.x, DIRECT);
62 PID myPIDy(&acc.y , &output.y, &setpoint.y, Kp.y, Ki.y, Kd.y, DIRECT);
63 PID myPIDz(&theta , &output.z, &setpoint.z, Kp.z, Ki.z, Kd.z, DIRECT);
64
65 int motor[4] = {1000,1000,1000,1000};
66
67 const byte interruptPin[4] = {53, 51, 49, 47};
68 const byte motorPin[4] = {9,10,11,12};
69 volatile int input[6] = {1000,1000,1000,1000,1000,1000};
70 volatile uint32_t t[7];
71
72 void setup() {
73   myServo[0].attach(motorPin[0]);
74   myServo[1].attach(motorPin[1]);
75   myServo[2].attach(motorPin[2]);
76   myServo[3].attach(motorPin[3]);
77   myServo[0].writeMicroseconds(motor[0]);
78   myServo[1].writeMicroseconds(motor[1]);
79   myServo[2].writeMicroseconds(motor[2]);
80   myServo[3].writeMicroseconds(motor[3]);
81   pinMode(interruptPin[0], INPUT_PULLUP);
82   pinMode(interruptPin[1], INPUT_PULLUP);
83   pinMode(interruptPin[2], INPUT_PULLUP);
84   pinMode(interruptPin[3], INPUT_PULLUP);
```

```
85    Serial.begin(115200);
86    Serial.println("REBOOT");
87
88    // Initialize the sensor (it is important to get calibration values
          stored on the device).
89    if (pressure.begin())
90      Serial.println("BMP180 init success");
91    else
92    {
93      // Oops, something went wrong, this is usually a connection p,roblem,
94      // see the comments at the top of this sketch for the proper
            connections.
95      Serial.println("BMP180 init fail (disconnected?)\n\n");
96      while(1); // Pause forever.
97    }
98    statusTemperature = pressure.startTemperature();
99    if (statusTemperature != 0)
100   {
101     delay(statusTemperature);
102     statusTemperature = pressure.getTemperature(T);
103     if (statusTemperature == 0)
104           Serial.println("error retrieving temperature measurement\n");
105     else
106     {
107           Serial.print(T);
108           Serial.println(" oC");
109           // Start a pressure measurement:
110           // The parameter is the oversampling setting, from 0 to 3 (
                highest res, longest wait).
111           // If request is successful, the number of ms to wait is
                returned.
112           // If request is unsuccessful, 0 is returned.
113           statusPressure = pressure.startPressure(3);
114           if (statusPressure != 0)
115           {
116               delay(statusPressure);
117               statusPressure = pressure.getPressure(P,T);
118               if (statusPressure == 0)
119                 Serial.println("error retrieving pressure measurement\n")
                      ;
120               else
121                 baseline = P;
122           }
123           else Serial.println("error starting pressure measurement\n");
124       }
125   }
126   else Serial.println("error starting temperature measurement\n");
127
128   Serial.print("baseline pressure: ");
129   Serial.print(baseline);
130   Serial.println("mbar");
131   Wire.begin();
132   lsm303.init();
```

```
133   lsm303.enableDefault();
134   lsm303.read();
135   N = (LSM303::vector<double>){lsm303.m.x-offset.x,lsm303.m.y-offset.y,
          lsm303.m.z-offset.z};
136   LSM303::vector_cross(&N,&acc,&E);
137   angle = atan2(E.y,E.x);
138   lm = sqrt(LSM303::vector_dot(&N,&N));
139   la = sqrt(LSM303::vector_dot(&lsm303.a,&lsm303.a));
140
141   // Try to initialise and warn if we couldn't detect the chip
142   if (!gyro.begin(gyro.L3DS20_RANGE_500DPS))
143   {
144     Serial.println("Oops ... unable to initialize the L3GD20. Check your
          wiring!");
145     while (1);
146   }
147   if(la>0){
148     kalAcc.x.set(lsm303.a.x/la);
149     kalAcc.y.set(lsm303.a.y/la);
150     kalAcc.z.set(lsm303.a.z/la);
151   }
152   if(lm>0){
153     kalMag.x.set(N.x/lm);
154     kalMag.y.set(N.y/lm);
155     kalMag.z.set(N.z/lm);
156   }
157   kalAcc.x.setQ(QAcc.x);
158   kalAcc.x.setQbias(QbiasAcc.x);
159   kalAcc.x.setRmeasure(RAcc.x);
160   kalAcc.y.setQ(QAcc.y);
161   kalAcc.y.setQbias(QbiasAcc.y);
162   kalAcc.y.setRmeasure(RAcc.y);
163   kalAcc.z.setQ(QAcc.z);
164   kalAcc.z.setQbias(QbiasAcc.z);
165   kalAcc.z.setRmeasure(RAcc.z);
166   kalMag.x.setQ(QMag.x);
167   kalMag.x.setQbias(QbiasMag.x);
168   kalMag.x.setRmeasure(RMag.x);
169   kalMag.y.setQ(QMag.y);
170   kalMag.y.setQbias(QbiasMag.y);
171   kalMag.y.setRmeasure(RMag.y);
172   kalMag.z.setQ(QMag.z);
173   kalMag.z.setQbias(QbiasMag.z);
174   kalMag.z.setRmeasure(RMag.z);
175   //turn the PID on
176   myPIDx.SetOutputLimits(-Max.x,Max.x);
177   myPIDy.SetOutputLimits(-Max.y,Max.y);
178   myPIDz.SetOutputLimits(-Max.z,Max.z);
179   myPIDx.Init();
180   myPIDy.Init();
181   myPIDz.Init();
182   attachInterrupt(digitalPinToInterrupt(interruptPin[0]), blink1, RISING)
          ;
```

```
183   Serial.println("start");
184   timer = micros();
185   t[6] = timer;
186   while(micros()-timer<wait){
187     Read();
188   }
189 }
190 // ---- initialisation
191
192 void Read(){
193   if(Serial.available()>0){
194       char first = Serial.read();
195       if (first=='0'){
196         Kp.x = Serial.parseFloat();
197         Kp.y = Serial.parseFloat();
198         Kp.z = Serial.parseFloat();
199         Ki.x = Serial.parseFloat();
200         Ki.y = Serial.parseFloat();
201         Ki.z = Serial.parseFloat();
202         Kd.x = Serial.parseFloat();
203         Kd.y = Serial.parseFloat();
204         Kd.z = Serial.parseFloat();
205         out = Serial.parseInt();
206         myPIDx.SetTunings(Kp.x,Ki.x,Kd.x);
207         myPIDy.SetTunings(Kp.y,Ki.y,Kd.y);
208         myPIDz.SetTunings(Kp.z,Ki.z,Kd.z);
209       } else {
210         opticalX = Serial.parseFloat();
211         opticalY = Serial.parseFloat();
212       }
213    }
214 }
```

You can get the whole source code from here git://s55969da3.adsl.online.nl/drone/firmware.git.

3-3 Middleware

Beside an Arduino we have also installed a Raspberry Pi in our drone. In the Raspberry Pi we run an AI program which is written in Java (source code git://s55969da3.adsl.online.nl/java/AIRaspberry.git). A webcam is connected to Raspberry Pi. This webcam is used as a part of optical flow sensor. The Self Charging Station will guide the drone to land on its platform by using another two cameras. We are also able to control the drone manually with a radio controller.

3-4 Front End

We use a web interface to configure the drone through Raspberry Pi, because you can run it remotely with any computer or smart phone. We can also read out the sensors' values from the Arduino. I would like to build a websocket base web interface, because it has two ways

communication protocol. This interface will be used for the tunning of the drone. We use try and error technique to find the proper values for our PID controller and Kalman Filter.

3-5 Estimation of the Budget

The part list of this drone is depicted in table 3-1.

Table 3-1: A part list of drone I.

Components	Cost (in euros)	Weight (g)	Dimension (mm) or Comments
4 x Brushless Motors	4 x 17.89	4 x 49	
4 x ESC's	4 x 10.85	4 x 19	20 A
Frame	-	100	3D printing
Arduino Due	45	36	101.52 x 53.3
5 x Ultrasonic Sensors	5 x 2.95	-	HC-SR04
10DOF	5	-	
Battery	17.96	244	112 x 34 x 30
Raspberry Pi	30	100	100 x 50
3 x Webcams	2 x 10	10	20 x 20
Total	200	800	

We use a Gantt Chart to plan our project. The Gantt Chart of our first drone is depicted in figure 3-2.

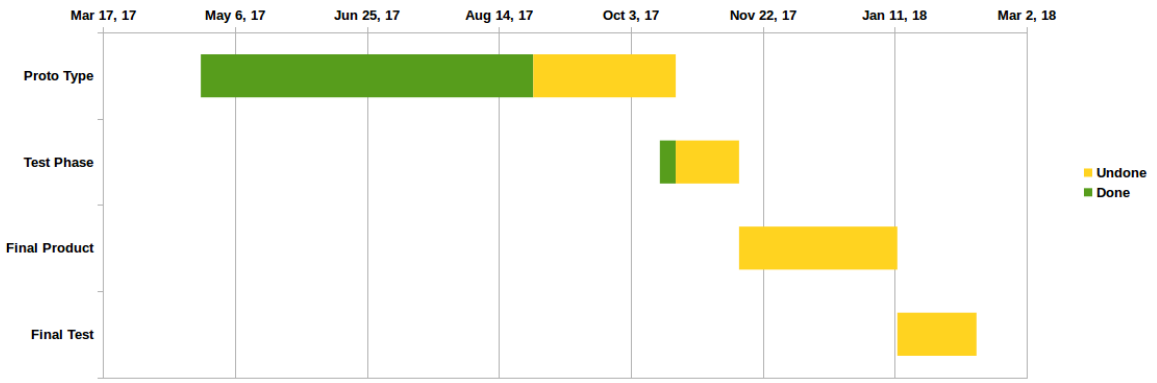


Figure 3-2: The Gantt Chart of this drone.

We expect to finish this drone within two years.

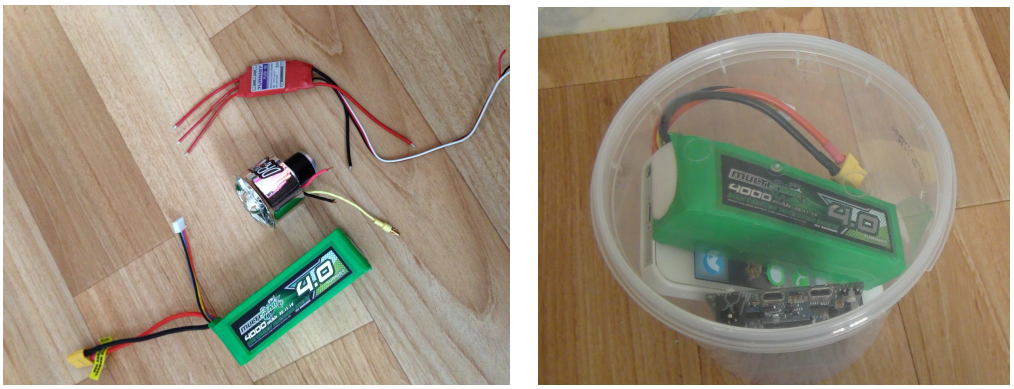
Chapter 4

Drone II

In this chapter we will explain our proposal of our second drone. In next sections we will explain the functions and the components of our drone. In the first section we discus about the design of the drone. Then we discus about the software of the drone. At the end we discus the budget and planning of the drone.

4-1 Design of the Drone II

This drone has four jet engines and four electronic speed controllers (ESC's). ESC's are controlled by an Arduino. We use the GPS from an iPhone to detect the location of our drone. We use an Ultrasonic Sensor to measure the height of the drone respect to the ground, and another four Ultrasonic Sensors to keep obstacle away in front of our drone. We use a gyroscope and an accelerometer to stabilize our drone in the air. We have a 4.0 Ah battery. We expect the fly time is about 5 to 10 minutes. For the future development we would like to have a self charging system for the drone. The drone will also be possible to dive into the water, but we first have to make it waterproof. An overview of the components is depicted in figure 4-1a and 4-1b. A 3D design of the frame of drone II is depicted in figure 4-2



(a) **(b)**

Figure 4-1: a) the battery, the jet engine and the ESC, and b) the body of drone II.

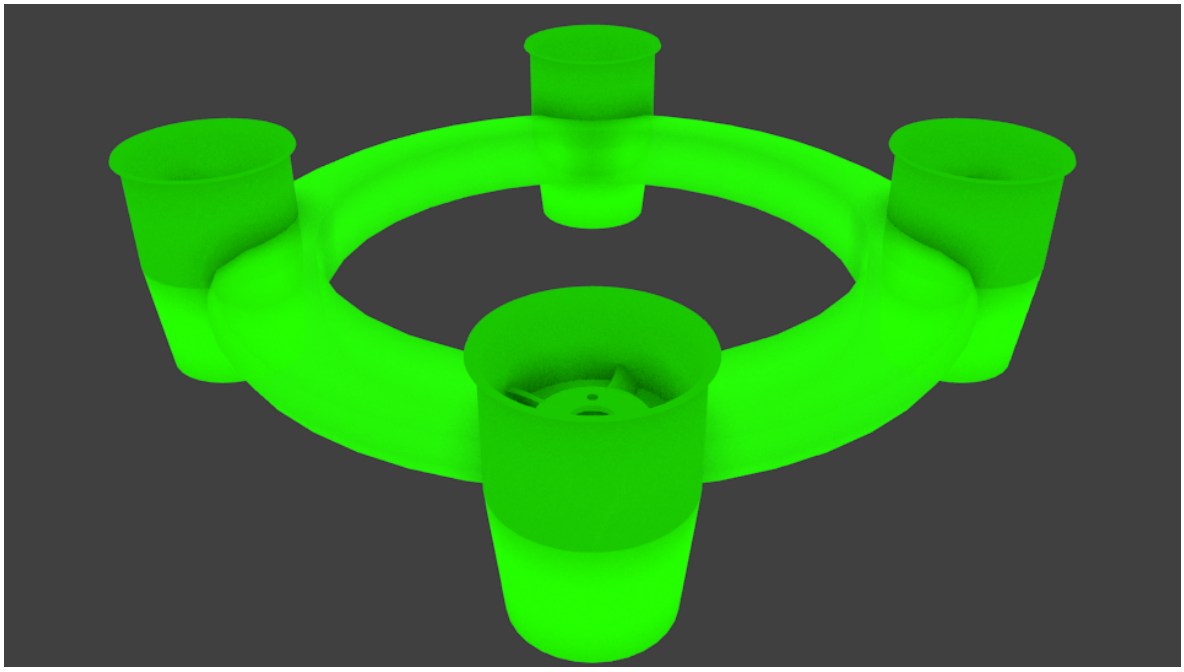


Figure 4-2: The frame of drone II.

4-2 Back End Software

An app, which is written in Objective C, is installed on the iPhone. Via WiFi the sensor data are sent to the Self Charging Station. The AI program, which is installed on the Self Charing Station, will process those data. The iPhone has a 11DOF sensor. It includes an accelerometer, a magnetometer, a gyroscope, a barometer and a GPS. The iPhone has also two cameras. One of them is used for the optical flow sensor. After processing the data we will send them to the Self Charging Station. The Self Charging Station has also two cameras. Those two cameras are used for vision location of the drone.

We have installed two Arduinos in our drone. One Arduino is connect to the radio receiver. Via the radio controller we send the instructions to the Arduino. Then it send to another Arduino. This Arduino controls the ESC's. The five Ultrasonic Sensors are also connected to this Arduino for distance detections.

4-3 Front End Software

We use the web interface to configure the Self Charging Station. It includes important parameters to control the drone. Those are parameters for PID controller and Kalman Filter. The difference between the first and the second drone is the first drone has everything inside the drone. The second drone has the controller unit on Self Charging Station.

4-4 Estimation of the Budget

The part list of this drone is depicted in table 4-1.

Table 4-1: A part list of drone II.

Components	Cost (in euros)	Weight (g)	Dimension (mm) or Comments
4 x Brushless Jet Engines	4 x 17.89	4 x 49	Dr. MadThrust 40 mm
iPhone 5	600	112	123.8 x 58.6 x 7.6
4 x ESC's	4 x 10.85	4 x 19	Turnigy Plush 18 A
Frame	-	100	3D printing
2 x Arduino Nanos	5	9	10 x 53.3
5 x Ultrasonic Sensors	5 x 2.95	-	HC-SR04
Battery	17.96	244	112 x 34 x 30
Radio Controller System	20	5	5 x 10
2 x Webcams	2 x 10	-	
Total	800	800	

The Gantt Chart of this drone is depicted in figure 4-3.

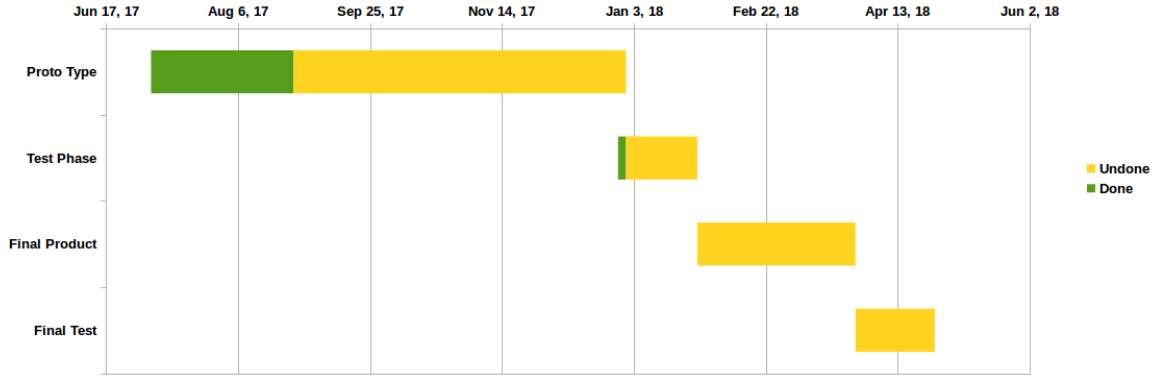


Figure 4-3: The Gantt Chart of drone II.

We also expect to finish this drone within two years.

Chapter 5

Conclusion

There is a big potential in the applications of a drone nowadays e.g. in agriculture, making area videos etc. It cost 500 euros to build a drone like this. The actual progress of this project can be download by https://s55969da3.adsl.online.nl/files/ganttchartdrone.ods. To build a swarm of drones is a challenging project, but we have to finish the AI drone first. To build a drone is easy, but the most difficult part is to find the proper parameters for the PID controller and Kalman Filter. It is a big achievement to build a drone like this. Drone I is a standard way to build a drone and Drone II is a challenging project.