UNIVERSITY OF IDAHO

CS CAPSTONE DESIGN

Capstone Portfolio

Drone Mission Planning Software

Team:
Mission Control

Authors:
David KLINGENBERG
Taylor TRABUN

Advisors:
Dr. Bruce BOLDEN
Dr. Robert RINKER

Customer:
Brandon ORTIZ

December 4, 2014

Contents

List of Figures

List of Tables

1　Team Member Contact Information

Name	Phone Number	Email Address
David Klingenberg	(208) 310-9657	bigwookiee@Gmail.com
Taylor Trabun	(509) 995-0904	trab1744@vandals.uidaho.edu

Table 1: Team Member Contact Information

2　Introduction

Software to create and upload a flight plan to a quad copter drone. The flight plan will be uploaded using xBee radio communication.

This project will use off-the-shelf parts. ATMEL$^{©}$ based microcontrollers found on ardunio based open source boards is the current preference.

2.1　Target Priorities

Number	Category	Need	Importance
1	Quadcopter	Center of Gravity Refined	5
2	Quadcopter	Reliable Flight	5
3	Quadcopter	Functioning xBee Hardware	4
4	Quadcopter	Hardware (Microcontroller) with xAPI and services to control flight	5
5	Quadcopter	Controlled with XP communications	4
6	Quadcopter	Autoland	5
7	Software	software package for flight planning	2
8	Software	API for sending commands from computer	2

Table 2: Priorities

3   Initial Client Interview Transcript 9/10/14

Mentor/Client: Brandon Ortiz

3.1   Meetings

We will be having weekly meetings in Brandon's office on Thursdays at 3:30 PM. These meeting will include status updates, further work on designs, troubleshooting, and assignment of tasks

3.2   End Goal

To have a stable and flying quadcopter that can be communicated with remotely. In addition, work done on a flight planning software (including GUI) should be underway. The project will be done in small steps, as this project requires research and development throughout.

3.3   First Steps

- Learn how quadcopter works
- Reconstruct quadcopter to be stable
- Learn how to fly quadcopter
- Understand flight computer documentation
- Design communications
- Be sure to use xAPI

3.4   Requirements

- Functional quadcopter (stable)
- Documentation of quadcopter construction
- Use of xAPI on arduino communication system
- Communication system using xBEE to communicate from computer to quadcopter
- Ability to send commands to quadcopter
- Flight planning software, including GUI

3.5   Other Notes

Other notes from the meeting included aviation terminology, how to pair the remote control and quadcopter receiver, quick tour of controller and motor adjustments, and a quick tour of flight computer.

4 Meeting Agendas

4.1 Sept. 10, 2014

Mission Control Team Agenda

Friday September 10, 2014.
1500 — 1600 in JEB Think Tank.

Type of Meeting
  Initial client interview.

Attendees
  David Klingenberg
  Taylor Trabun
  Brandon Ortiz

Topics

Topic	Responsible	Time (in minutes)
Product Overview	Brandon	15
System Requirements	Brandon	15
Tasks Breakdown	Open Discussion	15
Question & Answers	Open Discussion	25

Additional Information: This is our initial client interview.

4.1.1 Minutes from Friday September 10 Meeting

Refer to Section 3 initial client transcript.

4.2 Sept. 18, 2014

Mission Control Team Agenda

Thrusday September 18, 2014.
1500 — 1600 in JEB Think Tank.

Type of Meeting
  Initial Planning

Attendees
  David Klingenberg
  Taylor Trabun
  Brandon Ortiz
  Bruce Bolden

Topics

Topic	Responsible	Time (in minutes)
Progress Report	David, Taylor	5
System Overview	Brandon	10
Tasks Breakdown	Open Discussion	20
Additional Words of Wisdom	Bruce	5
Question & Answers	Open Discussion	20

Additional Information:

The rerouting and reconfiguring of the drone is proceeding nicely. It progress will be shown at the meeting time.

4.2.1   Minutes from Thursday September 18 Meeting

- 1505Meeting Started

- Discussed drone rebuild progress.

- Evaluated ESC bin for the drone.

  - Refer to figur 5 in Appendix C

- Discussed, evaluated, and illustrated the communication sequence.

  - Refer to figur 2 in Appendix A

- 1610 Meeting

4.3   Sept. 25, 2014

Mission Control Team Agenda

Thrusday September 25, 2014. 1530 — 1630 in JEB 37

Type of Meeting
Status Report and Next Week Planning

Attendees
David Klingenberg
Taylor Trabun
Brandon Ortiz

Topics

Topic	Responsible	Time (in minutes)
Progress Report	David & Taylor	10
Demonstrations	David & Taylor	10
New Tasks	Open Discussion	20
Question & Answers	Open Discussion	20

Additional Information:

4.3.1   Minutes from Thursday September 25 Meeting

- 1530 Meeting Start

- Discussed LCD use on Arduinos.

- Reviewed TUN packets.

- Status updates

    - Things moving along.
    - Getting closer to flying possibly next Thursday.

- xBee discussion on how to connect.

- Evaluated future problems.

    - Gyros and accelerometers need to be implemented separately from the flight computer.

- 1630 Meeting Ended

4.4   Oct. 2, 2014

Mission Control Team Agenda

Thrusday October 2, 2014.
1530 — 1630 in JEB 37

Type of Meeting
Status Report and Next Week Planning

Attendees
David Klingenberg
Taylor Trabun
Brandon Ortiz

Topics

Topic	Responsible	Time (in minutes)
Progress Report	David & Taylor	10
Demonstrations	David & Taylor	10
New Tasks	Open Discussion	20
Question & Answers	Open Discussion	20

Additional Information:

4.4.1   Minutes from Thursday October 2 Meeting

- 1530 Meeting Start

- Status updates.

  - Taylor has one-way communications working.
  - David finished a prototype for the ECS bin.
    * Bin needs its weight reduced.
    * ECS cables need to be lengthened.

- To

  - Taylor will attempt to get XP comm working.
  - David will finish quadcopter.
  - Get a new adrenal for running a second xBee radio.
  - Solder new LCD board.

- xBee Configuration notes.

  - Use XCTU tool for configuration.
  - Need FID drivers installed for XCTU tool.

- 1630 Meeting Ended

4.5   Oct. 9, 2014

Mission Control Team Agenda

Thrusday October 9, 2014.
1530 — 1630 in JEB 37

Type of Meeting
  Status Report and Next Week Planning

Attendees
  David Klingenberg
  Taylor Trabun
  Brandon Ortiz

Topics

Topic	Responsible	Time (in minutes)
Progress Report	David & Taylor	10
Demonstrations	David & Taylor	10
New Tasks	Open Discussion	20
Question & Answers	Open Discussion	20

Additional Information:

4.5.1   Minutes from Thursday October 9 Meeting

- 1530 Meeting Start

- Update

  - Taylor is preparing for snapshot day.
  - David
    * Quadcopter rebuilt.
    * Simple xBee terminals working between two computers.

- New Resources

  - UAV control paper with GUI design example.
  - Survey of UAV papers.

- Action Items

  - David will experiment with PWM and the quadcopter and portfolio.
  - Taylor will work on poster for snapshot day and continue working on communications.

- Test Flight

  - Quadcopter has severe drift forward. David will work on solution.

- 1630 Meeting Ended

4.6 Oct. 16, 2014

Mission Control Team Agenda

Thrusday October 16, 2014.
1530 — 1630 in JEB 37

Type of Meeting
 Status Report and Next Three Week Planning

Attendees
 David Klingenberg
 Taylor Trabun
 Brandon Ortiz

Topics

Topic	Responsible	Time (in minutes)
Progress Report	David & Taylor	10
Demonstrations	David & Taylor	10
New Tasks	Open Discussion	20
Question & Answers	Open Discussion	20

Additional Information:
 Our next meeting will be in 3 weeks Nov 6, 2014.

4.6.1 Minutes from Thursday October 16 Meeting

- 1530 Meeting Start

- Update

  - Tatlor reported on snapshot day and his progress with the zigBee radios.
  - David
    * Begin fine-tuning the drone for stabilization and self level flight. Drifting stability have been greatly improved.

- Action Items

  - David will continue to experiment with PWM and the quadcopter. He will explore control algorithms.
  - Taylor will continue his work on communications.

- Test Flight

  - Quadcopter severe forward drift has been improved. David needs to develop a battery frame to stop the batteries from shifting which is causing some of the uncontrolled drift.

- 1630 Meeting Ended

Mission Control Team Agenda

Thrusday November 6, 2014.
1530 — 1630 in JEB 37

Type of Meeting
Status Report and additional Short-term Planning.

Attendees
David Klingenberg
Taylor Trabun
Brandon Ortiz

Topics

Topic	Responsible	Time (in minutes)
Progress Report	David & Taylor	10
Demonstrations	David & Taylor	10
New Tasks	Open Discussion	20
Question & Answers	Open Discussion	20

Additional Information:
Our next meeting will be Nov 20, 2014.

4.7.1 Minutes from Thursday October 16 Meeting

- 1530 Meeting Start

- Update

  - Taylor
    * Gui mock-up finished, class documentation work (design review presentation, wiki).

  - David
    * Having a great deal of problem with PWM as an input to flight computer. Will have to try different firmware's for the flight computer and explore possible alternatives to PWM.

- New Resources

- Action Items

- David will continue to experiment with PWM and the quadcopter. Will explore control algorithms used by existing quad copters.

- Taylor will continue his work on communications.

- Test Flight

  - Quadcopter severe forward drift has been improved. David needs to develop a battery frame to stop the batteries from shifting.

- 1630 Meeting Ended

Mission Control Team Agenda

Thrusday November 20, 2014.
1530 — 1630 in JEB 37

Type of Meeting
Status Report and additional Short-term Planning.

Attendees
David Klingenberg
Taylor Trabun
Brandon Ortiz

Topics

Topic	Responsible	Time (in minutes)
Progress Report	David & Taylor	10
Demonstrations	David & Taylor	10
New Tasks	Open Discussion	20
Question & Answers	Open Discussion	20

Additional Information:
Our next meeting will be Dec 4, 2014.

4.8.1   Minutes from Thursday October 16 Meeting

- 1530 Meeting Start

- Update

  - Taylor

    * Worked with Brandon to debug XBee comms, still under-way Action Items 15min

  - David

    * Focusing more on senior design and plans on continuing to work on project next semester, code written for PWM flight control

- Action Items

  - David is working on PWM, PWM flight service, team citizenship form.

- Taylor is working on design document, wiki update, team citizenship form, continue working out XBee comm bugs and send EXTERNAL_LCD TUN packet to another Arduino successfully.

- Test Flight

  - Broken bones and foul weather will place any future flight testing on hold.

- 1630 Meeting Ended

5 Code

5.1 Supplemental quad copter autopilot V1.4

```
1  /****************************
2  Version 1.4
3
4  Supplemental quad copter Autopilot.
5
6   Contains a prototype controller algorithm
7   to maintain an altitude hold. In this
8   version   only an ultrasonic rangefinder
9   is used to measure altitude.
10
11  Important numbers for the kk2.1
12
13  center satick :1500 micro seconds
14  Full Right/Back: 2100
15  Full Left/Forward: 900
16
17  100% Throttle: 2300
18  0% Throttle: 1000
19  ****************************/
20
21  #include <Servo.h>
22  #include <NewPing.h>
23  #include <PID_v1.h>
24  #include <Adafruit_LSM303_U.h>
25  #include <Adafruit_BMP085_U.h>
26  #include <Adafruit_L3GD20_U.h>
27  #include <Adafruit_Sensor.h>
28  #include <Wire.h>
29
30
31  /*   INPUT RANGE */
32  #define ZERO_THROTTLE    1000
33  #define FULL_THROTTLE    2300
34
35  #define FULL_STICK_LEFT_FORWARD    900
36  #define FULL_STICK_RIGHT_BACK      2100
37  #define ZERO_STICK                 1500
38
39  /* Pin assignments. */
```

```
40 #define AILERONS_PIN     2
41 #define ELEVATOR_PIN     3
42 #define THROTTLE_PIN     4
43 #define RUDDER_PIN       5
44 #define AUX              6
45
46
47 /* Sonar Setup */
48 #define GROUND_PING_PIN              12
49 #define GROUND_ECHO_PIN              11
50 #define GROUND_MAX_SONAR_DISTANCE 200
51 #define GROUND_SONAR_ITERATION      5
52
53 /*  PID Setup */
54 #define thr_out_range 1.25
55 #define Kp_add        0
56 #define Ki_add        1
57 #define Kd_add        2
58 #define auxset_add    3
59 #define pidMode_add   4
60
61 void zero_stick();
62 void zero_all_inputs();
63 void set_thr(int);
64 void set_ali(int);
65 void set_elv(int);
66 void set_rud(int);
67 void setup_pins();
68 byte arm();
69 byte disarm();
70
71
72 /* Global Variables */
73 double                 altitude_hold, ground_range_value,
       throttle_position;
74 byte                   RangeTime_1cm;
75 int                    groundRangeTime;
76 Servo                  throttle, rudder, aileron, elevator, aux;
77 byte                   armed;
78 NewPing                ground_range(GROUND_PING_PIN, GROUND_ECHO_PIN,
    GROUND_MAX_SONAR_DISTANCE);
79
```

```
80  /* Global Sensor Variables */
81  Adafruit_LSM303_Accel_Unified acc =
        Adafruit_LSM303_Accel_Unified(30301);
82  Adafruit_LSM303_Mag_Unified    mag    =
        Adafruit_LSM303_Mag_Unified(30302);
83  Adafruit_BMP085_Unified         bmp   = Adafruit_BMP085_Unified(18001);
84  Adafruit_L3GD20_Unified         gyro  = Adafruit_L3GD20_Unified(20);
85  sensors_event_t event;
86
87  /* Global PID Variables */
88  PID altPID(&ground_range_value, &throttle_position, &altitude_hold, 2,
        5, 1, DIRECT);
89
90  /* Pin secttion. */
91  void setup_pins(){
92      aileron.attach(AILERONS_PIN);
93      elevator.attach(ELEVATOR_PIN);
94      throttle.attach(THROTTLE_PIN);
95      rudder.attach(RUDDER_PIN);
96  }
97
98  /*
99  void displaySensorDetails(void)
100 {
101     sensor_t sensor;
102
103     accel.getSensor(&sensor);
104     Serial.println(F("------------ ACCELEROMETER -----------"));
105     Serial.print   (F("Sensor:        ")); Serial.println(sensor.name);
106     Serial.print   (F("Driver Ver:   ")); Serial.println(sensor.version);
107     Serial.print   (F("Unique ID:    "));
            Serial.println(sensor.sensor_id);
108     Serial.print   (F("Max Value:     ")); Serial.print(sensor.max_value);
            Serial.println(F(" m/s^2"));
109     Serial.print   (F("Min Value:     ")); Serial.print(sensor.min_value);
            Serial.println(F(" m/s^2"));
110     Serial.print   (F("Resolution:    "));
            Serial.print(sensor.resolution); Serial.println(F(" m/s^2"));
111     Serial.println(F("------------------------------------------"));
112     Serial.println(F(""));
113
114     gyro.getSensor(&sensor);
```

```
115    Serial.println(F("------------- GYROSCOPE -------------"));
116    Serial.print   (F("Sensor:        ")); Serial.println(sensor.name);
117    Serial.print   (F("Driver Ver:    ")); Serial.println(sensor.version);
118    Serial.print   (F("Unique ID:     "));
          Serial.println(sensor.sensor_id);
119    Serial.print   (F("Max Value:     ")); Serial.print(sensor.max_value);
          Serial.println(F(" rad/s"));
120    Serial.print   (F("Min Value:     ")); Serial.print(sensor.min_value);
          Serial.println(F(" rad/s"));
121    Serial.print   (F("Resolution:    "));
          Serial.print(sensor.resolution); Serial.println(F(" rad/s"));
122    Serial.println(F("------------------------------------------"));
123    Serial.println(F(""));
124
125    mag.getSensor(&sensor);
126    Serial.println(F("----------- MAGNETOMETER -------------"));
127    Serial.print   (F("Sensor:        ")); Serial.println(sensor.name);
128    Serial.print   (F("Driver Ver:    ")); Serial.println(sensor.version);
129    Serial.print   (F("Unique ID:     "));
          Serial.println(sensor.sensor_id);
130    Serial.print   (F("Max Value:     ")); Serial.print(sensor.max_value);
          Serial.println(F(" uT"));
131    Serial.print   (F("Min Value:     ")); Serial.print(sensor.min_value);
          Serial.println(F(" uT"));
132    Serial.print   (F("Resolution:    "));
          Serial.print(sensor.resolution); Serial.println(F(" uT"));
133    Serial.println(F("------------------------------------------"));
134    Serial.println(F(""));
135
136    bmp.getSensor(&sensor);
137    Serial.println(F("--------- PRESSURE/ALTITUDE ----------"));
138    Serial.print   (F("Sensor:        ")); Serial.println(sensor.name);
139    Serial.print   (F("Driver Ver:    ")); Serial.println(sensor.version);
140    Serial.print   (F("Unique ID:     "));
          Serial.println(sensor.sensor_id);
141    Serial.print   (F("Max Value:     ")); Serial.print(sensor.max_value);
          Serial.println(F(" hPa"));
142    Serial.print   (F("Min Value:     ")); Serial.print(sensor.min_value);
          Serial.println(F(" hPa"));
143    Serial.print   (F("Resolution:    "));
          Serial.print(sensor.resolution); Serial.println(F(" hPa"));
144    Serial.println(F("------------------------------------------"));
```

```
145    Serial.println(F(""));
146
147    delay(1500);
148 }
149 */
150
151 /* Zero out the control x,y and rotational inputs. */
152 void zero_stick(){
153    aileron.writeMicroseconds(ZERO_STICK);
154    elevator.writeMicroseconds(ZERO_STICK);
155    rudder.writeMicroseconds(ZERO_STICK);
156 }
157
158 void zero_all_inputs(){
159    zero_stick();
160    set_thr(0);
161 }
162
163 /* Sets the throttle. Use a range of 0 to 100. */
164 void set_thr (int val){
165    throttle.writeMicroseconds(map(val, 0, 100, ZERO_THROTTLE,
          FULL_THROTTLE));
166    //Serial.println(map(val, 0, 100, ZERO_THROTTLE, FULL_THROTTLE));
167 }
168
169 /* Set the ailerons. Use a range of -100 to 100. */
170 void set_ail (int val) {
171    if (val == 0)
172      aileron.writeMicroseconds(ZERO_STICK);
173    else
174      aileron.writeMicroseconds(map(val, -100, 100,
          FULL_STICK_LEFT_FORWARD, FULL_STICK_RIGHT_BACK));
175 }
176
177 /* Set the elevator. Use a range of -100 to 100. */
178 void set_elv (int val) {
179    if (val == 0)
180      elevator.writeMicroseconds(ZERO_STICK);
181    else
182      elevator.writeMicroseconds(map(val, -100, 100,
          FULL_STICK_LEFT_FORWARD, FULL_STICK_RIGHT_BACK));
183 }
```

```
184
185 /*  Set  the   rudder.  Use  a  range  of  −100  to  100. */
186 void set_rud (int val) {
187   if (val == 0)
188     rudder.writeMicroseconds(ZERO_STICK);
189   else
190     rudder.writeMicroseconds(map(val, −100, 100,
            FULL_STICK_LEFT_FORWARD, FULL_STICK_RIGHT_BACK));
191 }
192
193 /* Arms the motors */
194 byte arm(){
195   set_thr (0);
196   set_rud(−100);
197   delay (1000);
198   set_rud (0);
199
200   return 1;
201
202 }
203
204 /* Disarms the motors */
205 byte  disarm(){
206   set_thr (0);
207   set_rud(100);
208   delay (1000);
209   set_rud (0);
210
211   return (0);
212 }
213
214 byte time_1cm(){
215   float temperature, time;
216   bmp.getTemperature(&temperature);
217   time = 2/((331.3 * sqrt(1 + temperature / 273.15))/10000);
218
219 //   Serial.print(temperature);
220 //   Serial.print(" C, time: ");
221 //   Serial.println(time,6);
222   return time;
223 }
224
```

```
225
226 void pid_loop(){
227  int thr_min, thr_max;    //
228
229  //thr_min = throttle_position − thr_out_range;
230  //thr_max = throttle_position + thr_out_range;
231  //altPID.SetOutputLimits(thr_min,thr_max);
232  altPID.Compute();
233  Serial.print((int)throttle_position);
234  throttle.writeMicroseconds((int)throttle_position);
235
236 }
237
238 void setup() {
239    Serial.begin(115200);
240
241    if(!acc.begin())
242    {
243      /* There was a problem detecting the ADXL345 ... check your
              connections */
244      Serial.println(F("Ooops, no LSM303 detected ... Check your
              wiring!"));
245      while(1);
246    }
247    if(!mag.begin())
248    {
249      /* There was a problem detecting the LSM303 ... check your
              connections */
250      Serial.println("Ooops, no LSM303 detected ... Check your wiring!");
251      while(1);
252    }
253    if(!bmp.begin())
254    {
255      /* There was a problem detecting the BMP085 ... check your
              connections */
256      Serial.print("Ooops, no BMP085 detected ... Check your wiring or
              I2C ADDR!");
257      while(1);
258    }
259    if(!gyro.begin())
260    {
261      /* There was a problem detecting the L3GD20 ... check your
```

```
            connections */
262    Serial.print("Ooops, no L3GD20 detected ... Check your wiring or
          I2C ADDR!");
263    while(1);
264    }
265
266    altPID.SetMode(AUTOMATIC);
267    altPID.SetOutputLimits(1000, 2300);
268    throttle_position = 1500;
269    altitude_hold = 20;
270    RangeTime_1cm = time_1cm();
271
272    setup_pins();
273    zero_all_inputs();
274    throttle.writeMicroseconds(throttle_position);
275    delay(1500);
276    //displaySensorDetails();
277 }
278
279
280 void loop(){
281
282    ground_range_value =
          ground_range.ping_median(GROUND_SONAR_ITERATION);
283    ground_range_value = (int)ground_range_value / RangeTime_1cm;
284
285    pid_loop();
286
287    acc.getEvent(&event);
288    Serial.print(" Acc Z: ");
289    Serial.print(event.acceleration.z);
290
291    gyro.getEvent(&event);
292    Serial.print(" gyro Z: ");
293    Serial.print(event.gyro.z);
294
295    Serial.print(" Ping: ");
296    Serial.print(ground_range_value);
297    Serial.println(" cm");
298
299
300
```

```
301    //armed = arm();
302
303    //delay (5000);
304
305    //armed = disarm();
306
307    //delay(10);
308 }
```

./Supplementary_autopilot.c

6 Design Presentation

6.1 Nov 13, 2014

See Next Page

Drone Mission Planning Software

David Klingenberg
Taylor Trabun

- Problem Definition
- Specific Component Design
  - ◦ Communications Design
  - ◦ Drone Design
  - ◦ Graphical User Interface Design
- Timeline
- Questions/Concerns

Overview

The goal of this project is to design and develop a graphical user interface (GUI) for drone mission planning.

Requirements:
- A user-friendly interface
- Allow 3-dimensional mission planning
- Upload the flight plan using XAPI and XBee
- Allow manual override
- Implement drone hardware for flight control

Problem Definition

The communication system for this project must allow for commands to be sent from a computer to a drone.

Requirements:
- XAPI and XBee hardware
- Specific TUN packets:
  ◦ Manual drone instructions (altitude, direction, takeoff, etc..)
  ◦ Settings
  ◦ Acknowledgement of packet received
  ◦ Heartbeat/status updates
  ◦ Override (manual, land)
  ◦ Flight plan protocol
    ▪ Initialize for upload
    ▪ Get instructions
    ▪ Echo instructions
    ▪ Terminate upload

Communication Design

Simplified Drone Communication Sequence

28

Drone Design
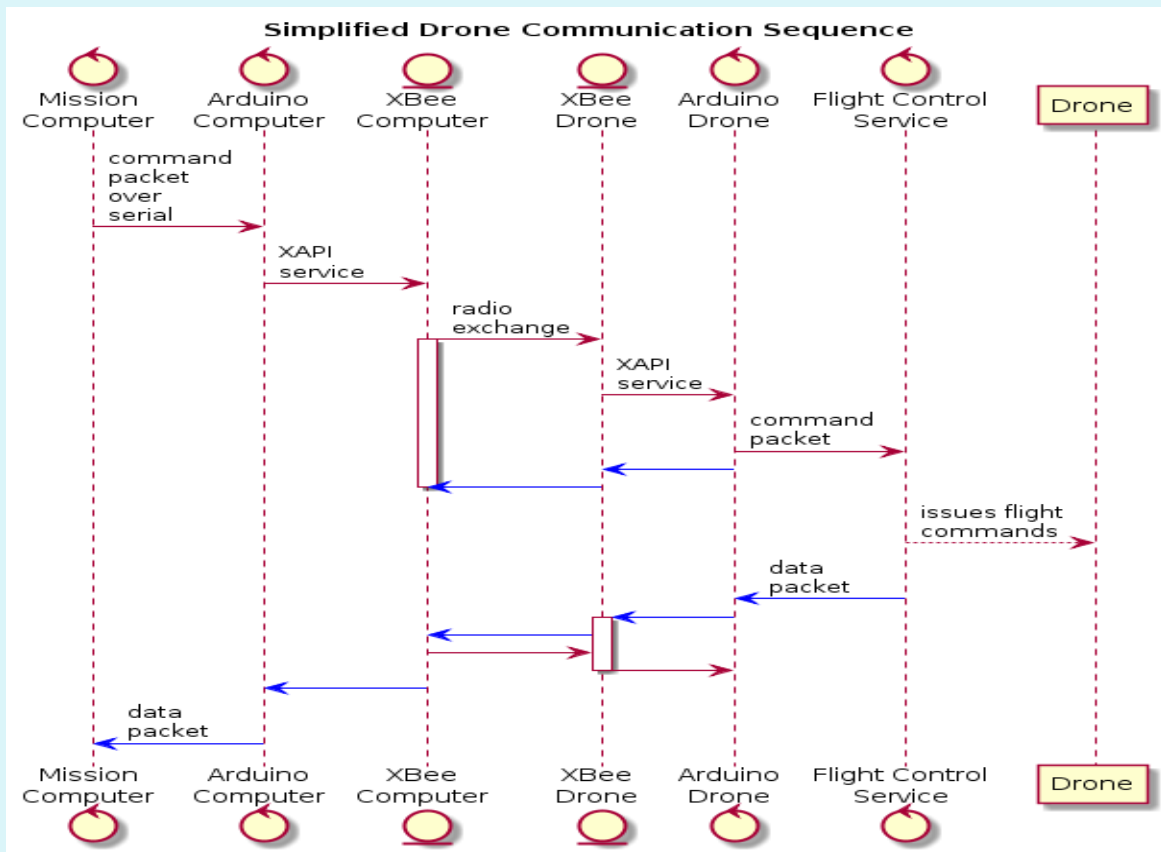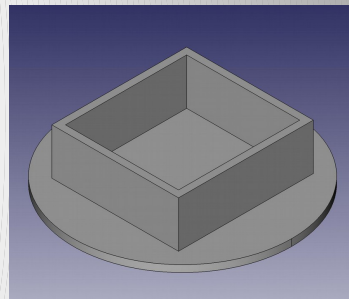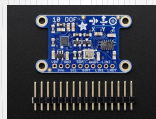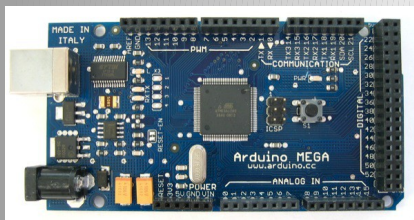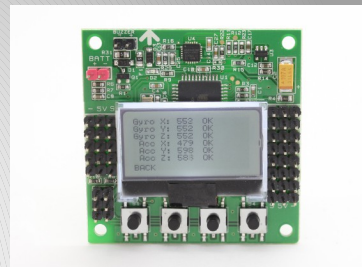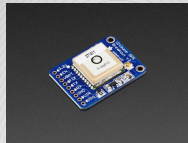
Graphical User Interface Design
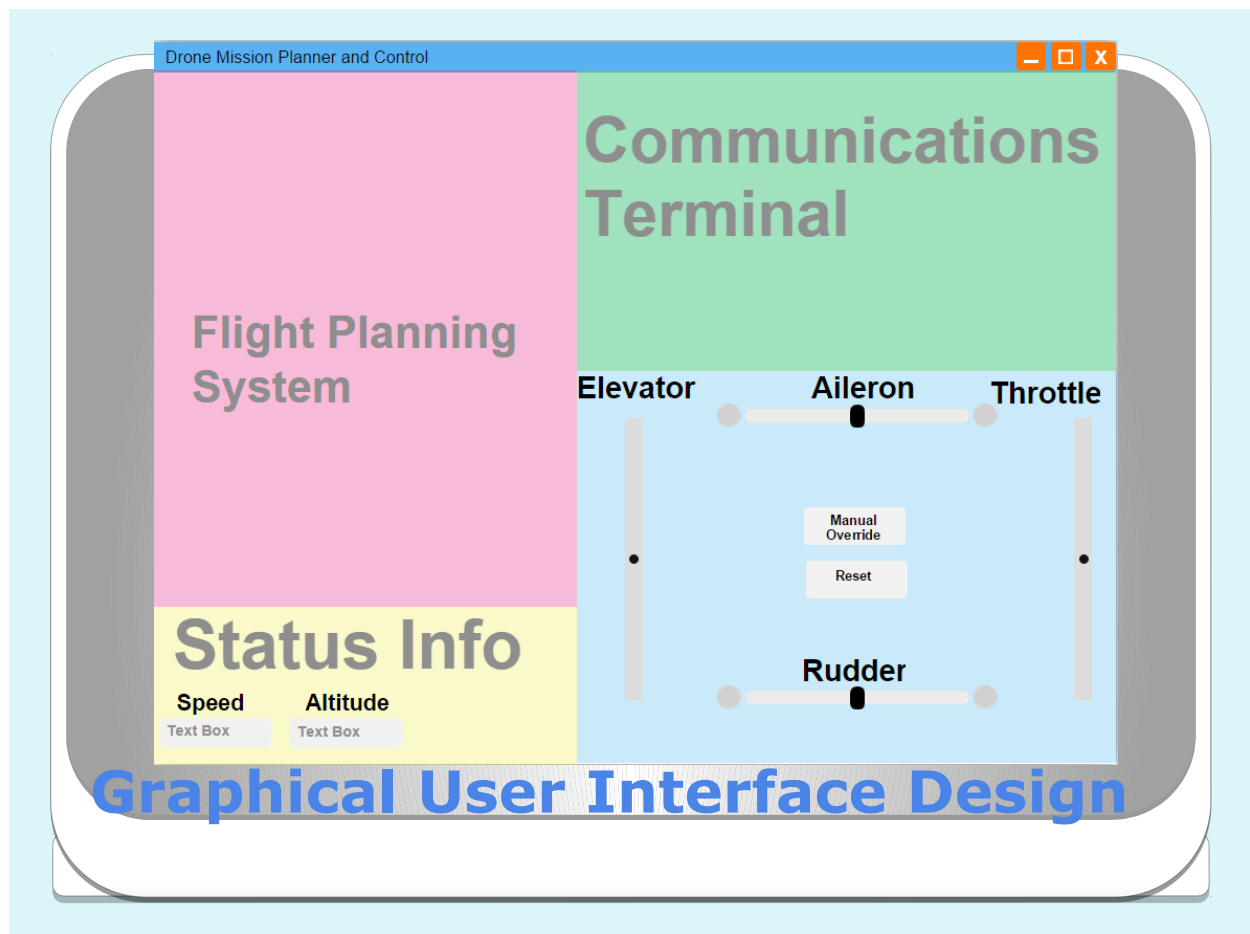
Re-design drone for reliable flight

Finish implementing communications between two Arduino boards over XBee using XAPI

Finish designing all needed packet types for drone control, status, ACK, and override.

Implement service on Arduino to send commands to flight computer

Finalize GUI design, implement communications system, design mission planning sub-system

Timeline

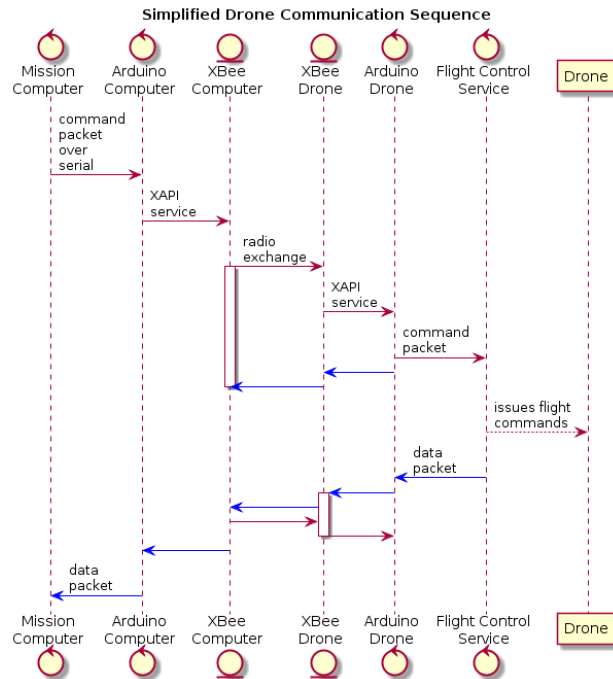# Appendices

A   Miscellaneous UML Charts

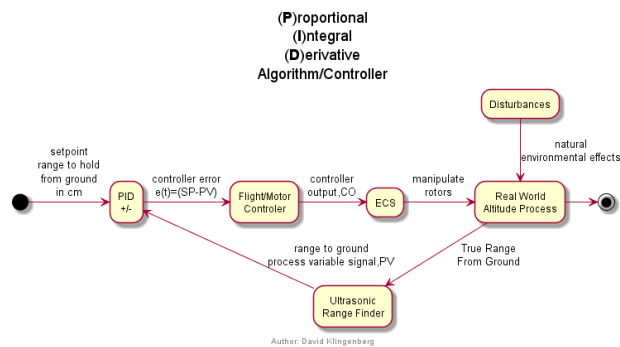Figure 1: Communication Sequence

Figure 2: PID Controller

B   ATMEL© Microcontrollers

Features

- High-performance, Low-power Atmel® AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 × 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20MHz
- High Endurance Non-volatile Memory segments
  - 64 Kbytes of In-System Self-programmable Flash program memory
  - 2 Kbytes EEPROM
  - 4 Kbytes Internal SRAM
  - Write/Erase cyles: 10,000 Flash/100,000 EEPROM[1][3]
  - Data retention: 20 years at 85°C/100 years at 25°C[2][3]
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel, 10-bit ADC
    - Differential mode with selectable gain at 1x, 10x or 200x
  - Byte-oriented Two-wire Serial Interface
  - One Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- Speed Grades
  - ATmega644V: 0 - 4MHz @ 1.8V - 5.5V, 0 - 10MHz @ 2.7V - 5.5V
  - ATmega644: 0 - 10MHz @ 2.7V - 5.5V, 0 - 20MHz @ 4.5V - 5.5V
- Power Consumption at 1MHz, 3V, 25°C
  - Active: 240µA @ 1.8V, 1MHz
  - Power-down Mode: 0.1µA @ 1.8V

Notes: 1. Worst case temperature. Guaranteed after last write cycle.
         2. Failure rate less than 1 ppm.
         3. Characterized through accelerated tests.

8-bit Atmel Microcontroller with 64K Bytes In-System Programmable Flash

ATmega644/V

2593O–AVR–02/12

Figure 3: ATmega644

34

Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V

8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash

Features

- High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 135 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 × 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16MHz
  - On-Chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 64K/128K/256KBytes of In-System Self-Programmable Flash
  - 4Kbytes EEPROM
  - 8Kbytes Internal SRAM
  - Write/Erase Cycles:10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/ 100 years at 25°C
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
    - Endurance: Up to 64Kbytes Optional External Memory Space
- Atmel® QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix acquisition
  - Up to 64 sense channels
- JTAG (IEEE® std. 1149.1 compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits (ATmega1281/2561, ATmega640/1280/2560)
  - Output Compare Modulator
  - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
  - Two/Four Programmable Serial USART (ATmega1281/2561, ATmega640/1280/2560)
  - Master/Slave SPI Serial Interface
  - Byte Oriented 2-wire Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 54/86 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)
  - 64-pad QFN/MLF, 64-lead TQFP (ATmega1281/2561)
  - 100-lead TQFP, 100-ball CBGA (ATmega640/1280/2560)
  - RoHS/Fully Green
- Temperature Range:
  - -40°C to 85°C Industrial
- Ultra-Low Power Consumption
  - Active Mode: 1MHz, 1.8V: 500µA
  - Power-down Mode: 0.1µA at 1.8V
- Speed Grade:
  - ATmega640V/ATmega1280V/ATmega1281V:
    - 0 - 4MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
  - ATmega2560V/ATmega2561V:
    - 0 - 2MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
  - ATmega640/ATmega1280/ATmega1281:
    - 0 - 8MHz @ 2.7V - 5.5V, 0 - 16MHz @ 4.5V - 5.5V
  - ATmega2560/ATmega2561:
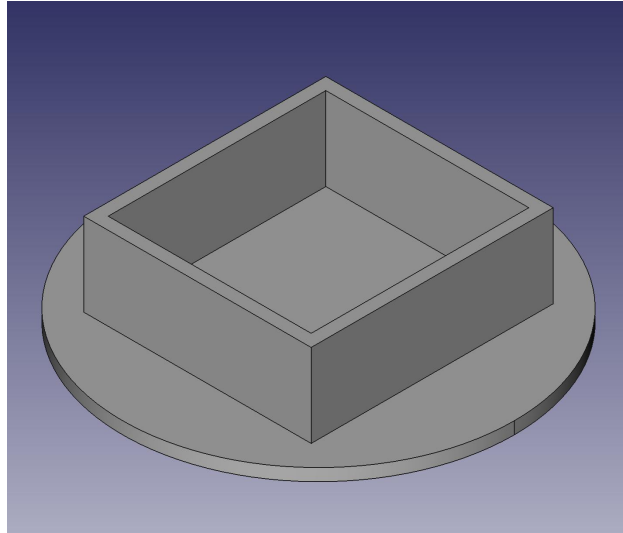   - 0 - 16MHz @ 4.5V - 5.5V

Figure 4: ATmega2560

35

C   Technical Drawings

Figure 5: Electronic speed controller part bin