VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Aleksi Olkkonen

Quadcopter flight controller

VAASAN AMMATTIKORKEAKOULU

Tietotekniikka
2018

TIIVISTELMÄ

Tekijä	Aleksi Olkkonen
Opinnäytetyön nimi	Quadcopter flight controller
Vuosi	2018
Kieli	Englanti
Sivumäärä	x
Ohjaaja	Jani Ahvonen

Opinnäytetyön tarkoituksena oli suunnitella, rakentaa ja ohjelmoida nelikopteri. Nelikopteri rakennettiin valmiista komponenteista ja mikrokontrollerille, antureille ja muille laitteille suunniteltiin ja rakennettiin piirilevy. Ohjain rakennettiin käyttäen potentiometrejä, mikrokonterolleria Työhön kuuluu ohjaimen ja nelikopterin välinen kommunikaatio, nelikopterin asennon laskeminen käyttäen kiihtyvyysanturia, gyroskooppia ja magnetometriä

Avainsanat Nelikopteri, sulautetut järjestelmät, säätötekniikka, LoRa

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Bachelor of engineering (IT)

ABSTRACT

Tekijä Aleksi Olkkonen
Opinnäytetyön nimi Quadcopter flight controller
Year 2018
Language English
Pages x
Ohjaaja Jani Ahvonen

Scope of this thesis is to design, build and program a quadcopter. Building a quadcopter requires knowledge of quadcopter's physical requirements and choosing compatible components such as battery, electronic speed controllers and motors. A circuit board is designed to connect microcontrollers, sensors and other devices together and provide supply voltage for them. A custom joystick is b

Flight controller is a program that reads user input and sensor measurements to calculate current orientation and target orientation.that uses sensor measurements to estimate quadcopter's orientation and controls motor speed based on PID controller output. Joystick communicates to the quadcopter using LoRa technology and part of the thesis is to find out if it is suitable technology for this use case.

Keywords Quadcopter, embedded systems, control theory, LoRa

TIIVISTELMÄ
ABSTRACT
INDEX OF FIGURES
INDEX OF TABLES

TABLE OF CONTENTS

INDEX OF FIGURES

INDEX OF TABLES

1 TERMS AND ABBREVIATIONS

BEC	Battery Elimination Circuit
(C)CW	(Counter) Clockwise
Drone	A general name for different type of multicopeters
ESC	Electronic Speed Controller
FPV	First Person View
IMU	Inertial Measurement Unit
Kv	rpm/V – rounds per minute per volt
LiPo	Lithium Polymer battery
LoRa	Long Range wireless communication technology
PDB	Power Distribution Board
Pitch	Rotation around X-axis
PWM	Pulse Width Modulation
Roll	Rotation around Y-axis
SPI	Serial Peripheral Interface
Yaw	Rotation around Z-axis

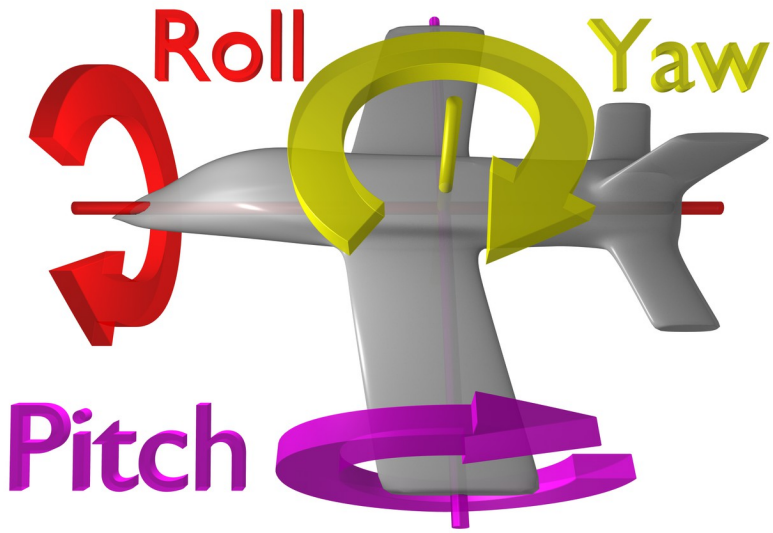
2 REQUIREMENT SPECIFICATION

3 FLYING THE QUADCOPTER

3.1 FLIGHT MODES

Quadcopters usually support at least two flight modes: Acro and self-level mode. In this thesis only self-level mode is implemented, which is easier to fly, but harder to implement.

A quadcopter can rotate on X, Y and Z axes. Rotation around X-axis is called pitch, rotation around Y-axis is called roll and rotation around Z-axis is called yaw.



Drawing 1: Pitch, roll and yaw [0]

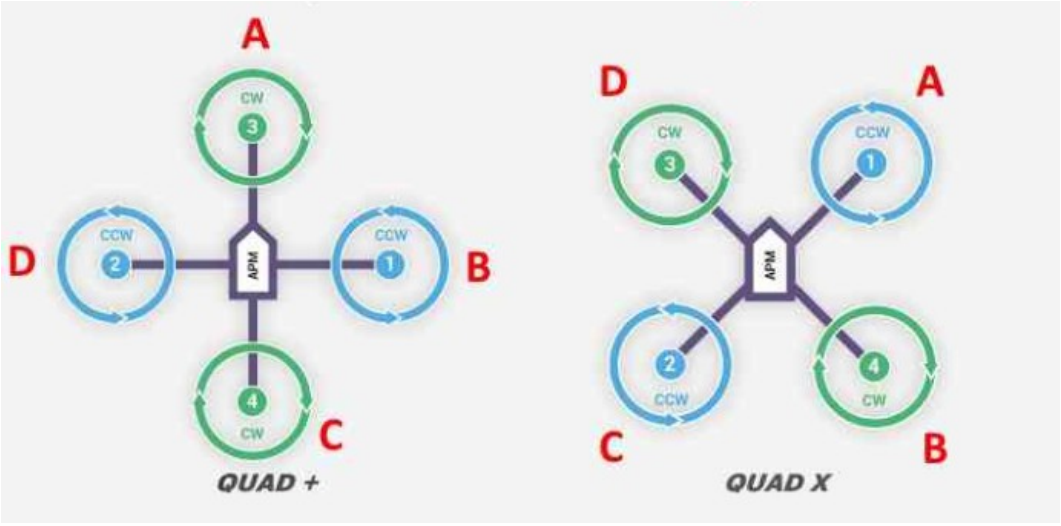
In acro mode pitch, roll and yaw control angular velocity of the quadcopter. Angular velocity is measured with gyroscope in degrees per second on each axis. User input will increase or decrease quadcopter's angular velocity and the person flying the quadcopter needs to be quite good at controlling the quadcopter, or it can come crashing down.

In self-level mode user input controls quadcopter's orientation directly. If the joystick is in middle, the flight controller will attempt to bring quadcopter into neutral orientation. Turning joystick to some direction will change quadcopter's target orientation on that axis.

3.2 QUADCOPTER MANEUVERS

A quadcopter has four different maneuvers that the user can control with left and right joysticks of the controller. Maneuvers are achieved by spinning motors at different speeds.

There are two different configurations for a quadcopter: a plus and a X configuration. A plus configuration has one motor on each side. X configuration has two front motors and two back motors. Shape of the Spendix S250Q frame does not allow implementing a plus configuration, because motors are closer to each other on Y-axis than they are on X-axis.



Drawing 2: Quadcopter flight configurations

Following table describes motor control in X configuration.

Command	Joystick	Motor speed	Maneuver
Throttle	Left joystick Y-axis	Each motor spins at equal speed	Increases or decreases altitude
Roll	Right joystick X-axis	Motors A and B spin at different speed than motors D and C	Move left or right
Pitch	Right joystick Y-axis	Motors A and D spin at different speed than motors B and C	Move forwards or backwards
Yaw	Left joystick X-axis	CW motors spint at different speed than CCW motors	Rotate left or right around drone's local Z-axis

Table 1: X configured quadcopter maneuvers

4 HARDWARE

4.1 COMPONENT LIST

Below is a list of most important hardware components used in this project. The total price of all components is 338,07 €. Most of the purchases were kindly supported by VAMK, University of Applied Sciences.

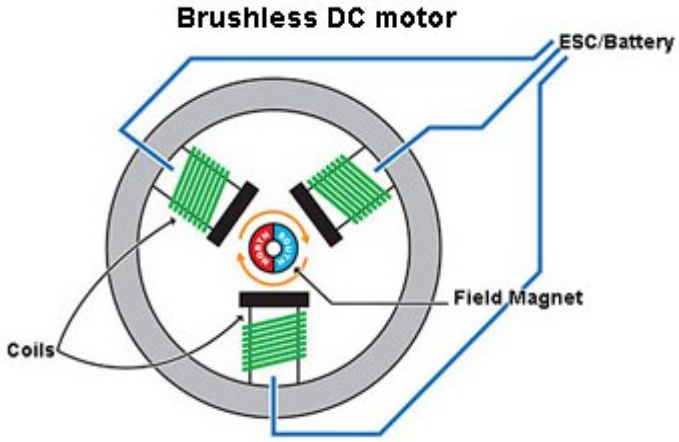
Part	**Description**	**Price (€)**
Motors	2x CCW and 2x CW EMAX MT2204	68
Electronic speed controllers	4x Emax 20A ESC with BEC	44
Propellers	4x 6"4 propellers	4,4
Battery	2x Gens ace 2200 mAh 3 cell, 11.1 V, 25C LiPo battery	59,8
Charger	GT Power B3 2-3s LiPo charger	14
Microcontroller	2x Adafruit feather 32u4 RFM95	57,32
Sensors	PmodNAV 9-axis IMU	27,13
Frame	Spedix S250Q	26,13
Circuit board	Custom made circuit board for PmodNav, microcontroller and connections	12,3
Power distribution board	4 Axis Electric Connection Plate Quad Power Distribution Board	5,7
PWM module	Adafruit 8-channel servo PWM module	11,2
Joystick battery	ValueLine power bank 2200 mAh	8,09
		338,07

Table 2: List of components

4.2 MOTORS

A quadcopter needs two clockwise motors and two counter clockwise motors. Having motors that spin different directions and are positioned properly will eliminate torque in one direction which would cause the quadcopter to yaw to opposite direction.

Brushed and brushless DC motors are two common types of motors used in quadcopters. Brushless motors are more popular because they are smaller, offer more power, higher efficieny and less mechanical wear. However they are more expensive and require an electronic speed controller.



Drawing 3: Brushless DC motor [0]

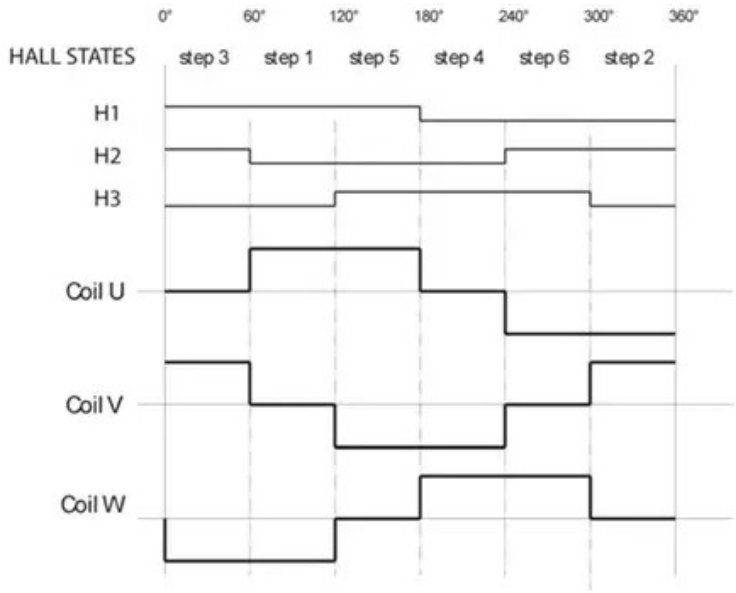
In a 3-phase brushless DC motor electronic speed controller controls the motor by rapidly switching current input for each of the coils. This creates a rotating magnetic field around the coil and spins the magnet. Magnet is attached to the same shaft as propeller and causes it to spin.

Motors with lower Kv rating produce higher torque and are able to spin a bigger propellers. Racing size quadcopters use small propellers, motors with higher Kv rating and therefore they maneuver faster but can lift less mass. Kv is equivalent to rpm/V, that is rounds per minute per voltage when there is no load attached to the motor. The more input voltage is supplied to the motor the faster it spins. Emax MT2204 motors are used in this project. Their Kv rating is 2300. Drawing full voltage output from a 11.1 V battery results into 25 530 rpm with no load attached.In practice their rpm is slower due to drag and proteries of the propellers being used.

4.3 ELECTRONIC SPEED CONTROLLERS

Electronic speed controllers control current flow from the battery to the motors according to PWM signal. PWM signal determines the voltage and current flow to the coils which affects the speed and torque of motors. Hall-sensors are used to determine position of the rotor and accurately control rotation of the magnetic field by energizing three coils of a brushless motor in specific order.

Figure below shows how coils U, V and W are energized in different phases according to the states of Hall-sensors H1, H2 and H3. Each step one coil is in neutral state, one in positive state and one in negative state. This forces motor to spin into specific position. Next step coils are energized in different order and motor spins to next position. Total of 6 steps are required to spin the motor 360 degrees.



Drawing 4: 3-phase brushless DC motor steps [0]

ESC's maximum current rating must be equal or higher than maximum current draw for the motor. Emax BLHeli 20 A ESC has high enough current rating for Emax MT2204 motor that requires 11.5 A at maximum thrust. It also has a built in battery elimination circuit that can be used to get a supply voltage for the microcontroller and other devices in the quadcopter. Emax BLHeli ESC provides 5 V / 2 A supply voltage.

Most ESC firmwares can be configured with parameters such as brake type, motor timing mode, motor control frequency and low voltage protection. These will affect flight properties of the quadcopter. Configuration can be done with the joystick or using an ESC programming toolkit and software.

4.4 BATTERY

Emax MT2204 motor requires 11.5 A current at maximum thrust. Spinning all four motors at maximum thrust requires 46 A current output from the battery. Gens Ace 3S LiPo battery is used in this project. It has 3 cells, each having nominal voltage of 3.7 V, and that results into total of 11.1 V. Its 2200 mAh capacity provides roughly 5-12 min flight time.

LiPo batteries are used because they have high discharge rate and are relatively compact in size. Battery's discharge rate, C, describes how fast it can discharge and how much current it can output. If battery's capacity is 2200 mAh and discharge rate is 1 C, the battery will deliver its capacity in 1 hour outputting 2,2 A. Therefore battery used in this project has discharge rate of 25, so it can output atleast 46 A required by the motors. 2200 mAh capacity with discharge rate of 25 can output 55 A and can discharge in 2,4 minutes. However most of the time the motors don't draw maximum current and estimated flight time is few minutes

longer.

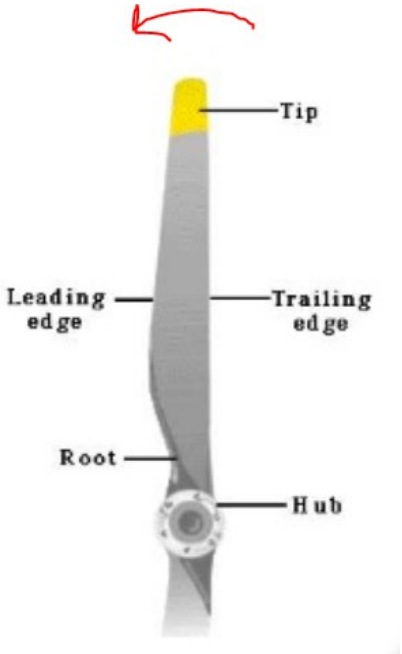
4.4.1 Safety

User needs to be careful when charging, discharging and storing a LiPo battery because improper use of the battery can be dangerous. Damaged battery can start a fire and it burns at hot temperatures. Battery should be in a LiPo safety bag when it is being stored or charged. If there is visible expanding on battery cells or it does not charge, the battery should be disposed properly. Battery charger should be high quality with indicators when the battery is fully loaded and detection to stop charging when the batteries are fully charged or damaged.

4.5 PROPELLERS

Motor's datasheet is the best reference for choosing an optimal propeller for the motor. Propeller length must be also suitable for the quadcopter's frame size so that the propeller tip does not get too close to another propeller or the frame. Propeller length is measured from tip to tip. Emax MT2204 motor is designed for 5 to 6 inch propellers. A longer propeller or a propeller with higher pitch produces more thrust but requires more torque. Pitch determines the angle between leading and trailing edge.

It is also important to choose a propeller according to motor direction. When the motor is spinning, leading edge of the blade must be leading the trailing edge to the spin direction. Leading edge is higher than trailing edge and therefore the propeller pushes air downwards and lifts the quadcopter in the air.



Drawing 5: Propeller [4]

There are propellers that have either two or three blades. Propellers with two blades are more efficient than the ones with three. They require less torque but produce less thrust and therefore do not maneuver the quadcopter as fast.

4.6 MASS / THRUST RATIO

Emax MT2204 datasheet defines that the maximum thrust for 6”3 propeller is 440 g. All four motors combined the thrust is 1760 g. As a rule of thumb maximum mass of the quadcopter should be half of the total thrust, but the mass can be even lower to achieve faster maneuverability. Mass of the built quadcopter is roughly 650 g which is lower than the suggested maximum of 880 g.

4.7 SENSORS

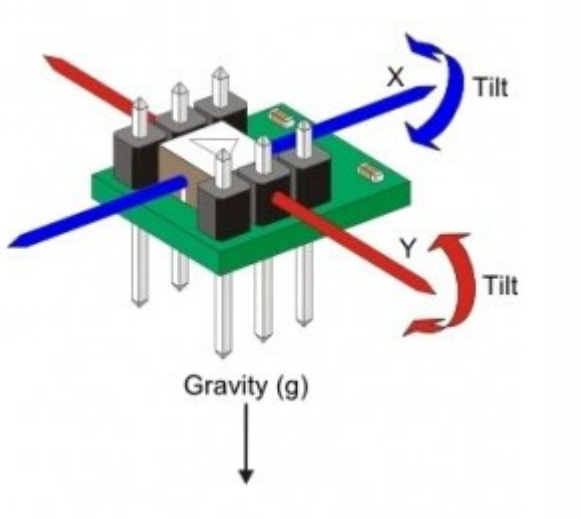
Different types of sensors are required to estimate quadcopters orientation. A self-leveling mode requires an accelerometer, a gyroscope and a magnetometer. Other sensors can be used to provide various types of measurements such as atmospheric pressure, which can be used to maintain the altitude.

PmodNav is used in this thesis. It has a LSM9DS1 chip that contains 3-axis accelerometer, gyroscope and magnetometer. Additionally it has a LPS25HB chip that contains a barometer.

4.7.1 Accelerometer

Accelerometer provides linear acceleration on 3 axes in g. Value of g is 9.81 m/s^2 and it is equivalent to acceleration caused by earth's gravitational force. Due to that force the accelerometer measures 1 g when it is not moving. If sensors orientation is neutral, 1 g is measured on Z-axis. Once it rotates around X or Y axis, acceleration can be measured also on other axes. From this data it is possible to calculate accelerometer's pitch and roll; rotation around X and Y axes.

Rotation around axis that is pointing to the same direction as gravitational force vector cannot be measured. For example in neutral orientation Z-axis is pointing to the same direction as gravitational force vetor. Rotating around Z-axis does not affect acceleration measured X or Y axes and therefore results to 1 g on Z-axis.



Drawing 6: Accelerometer
https://www.iotone.com/term/accelerometer/t20

quadcopter's orientation can be calculated from accelerometer data using following equations:

```
pitch = atan(accelerometer.Y, accelerometer.Z) * 180 / Pi
roll = atan(accelerometer.X, accelerometer.Y) * 180 / Pi
```

Rotation around X-axis is called pitch and it can be measured on Y-axis. Rotation around Y-axis can be measured on X-axis and it is called roll. Both values are converted from radians to degrees.
Accelerometer is easly affected by noise, such as quadcopter's motors causing the frame shake. Therefore it is important that the accelerometer is installed on a noise reduction material such as rubber, and is used together with gyroscope and complementary filter.

4.7.2 Gyroscope

Gyroscope measures angular velocity on 3 axes in degrees per second. quadcopter's orientation cannot be calculated from a single sensor measurement like it can be with accelerometer. When quadcopter has no angular velocity on an axis, gyroscope measures 0 dps. When quadcopter starts rotate around an axis, gyroscope will measure the angular velocity and it will be integrated by the microcontroller to estimate quadcopter's orientation:

```
pitch = pitch + gyroscope.X * deltaTime
roll = roll + gyroscope.Y * deltaTime
yaw = yaw + gyroscope.Z * deltaTime
```

This way gyroscope can be used to measure offset from the original rotation around an axis. Theoritically original rotation is 0 degrees pitch and roll when the

quadcopter is placed on a flat surface. Delta time is used as a multiplier because quadcopter flight controller loop executes n amount of times in a second, and measured unit is degrees per second. The faster the angular velocity can be measured and integrated the more accurate result is. However there will be always minor inaccuracies due to sensor's refresh rate, measurement accuracy and floating point operations. This will result into drift meaning that when quadcopter rotates away from the original rotation and retuns back to the original rotation, the calculated orientation will have some offset from the original position. When integration is run for longer time these inaccurasies can increase and decrease.

Gyroscope is not prone to noise caused by oscalliation of the quadcopter's frame and therefore it is used to compensate noisy measurements of an accelerometer.

4.7.3 Magnetometer

Magnetometer measures earth's magnetic field in G (gauss) and it is used to calculate quadcopter's geographical heading. Magnetometer readings are converted into polar coordinates which behave similar to a compass. Polar coordinate measures rotation around quadcopter's Z-axis in 360 degrees, where north is 0 degrees and south is 180 degrees. Magnetometer helps to compensate gyroscope's drift when calculating yaw for the quadcopter.

4.7.4 Barometer

Barometer measures atmospheric pressure in hPa (hectopascal). When the barometer is properly calibrated and has accurate resolution it can measure even small changes in air pressure when quadcopter's altitude changes and therefore it is used to maintain altitude.

However changing weather conditions can change atmospheric pressure drastically and in that case barometer's measurement will drift from the original altitude and maintaining the altitude is not accurate. Barometer can provide accurate altitude information only short term. Rangefinder would provide much more accurate measurement of quadcopter's altitude relative to ground level. It can measure distance to ground and therefore it can be used to adjust altitude even when terrain height changes.

4.7.5 Sensor configuration

LSM9DS1 chip's sensors can be configured for different measurement ranges and refresh rates. Refresh rate is configured to its maximum of 952 Hz to provide fresh measurement data for the flight controller as often as possible. Each axis outputs 16-bit value to represent the measured value. Larger measurement range leads to decreased precision because capacity of a 16-bit variable is limited.

Each sensor is configured to use the smallest measurement range to achieve the best resolution. At least for gyroscope the range should be more than enough, because the quadcopter would need to maneuver a 360 degree flip around one axis to get a measurement over 245 dp. High resolution is preferred when attempting

to estimate quadcopter's position correctly. However, different sensor measurement ranges should be tested to verify how the configuration affects flight performance of the quadcopter. Following table shows all available measurement range configurations for the sensors.

Sensor name	**Measurement range**
Accelerometer	±2, ±4, ±8 and ±16 g
Gyroscope	±245, ±500, ±2000 dps
Magnetometer	±4, ±8, ±12, ±16 gauss

Table 3: Sensors' measurement ranges

LPS25HB barometer measurement range is from 260 to 1260 hPa. It can run in high resolution or low power consumption mode and provides measured values in 24-bits.

Configuring both chips and reading measurements can be done by using either I2C or SPI interfaces. Open source library is available online and it allows easy usage of the sensors on the PmodNav IMU.
https://reference.digilentinc.com/reference/pmod/pmodnav/reference-manual

4.8 MICROCONTROLLER

The quadcopter and joystick uses an Adafruit Feather 32u4 that has a LoRa RFM95 module preinstalled. This was the most simple way to establish long range wireless communication between both devices. Microcontroller is ATMega 32u4 8-bit AVR clocked at 8 MHz. It has 3.3 V logic level, 32 KB program memory and 2.5 KB SRAM. The chip can be programmed using a USB 2.0 interface. ATMega 32u4 has also other features required by flight controller and joystick:

- 10x analog I/O pins for reading potentiometers or battery state
- Hardware SPI for reading and writing to sensors and LoRa module
- Hardware PWM generators to control signal for ESCs
- External interrupts and timer interrupts
- It is as light as a feather – mass of 5.5 grams adds nearly nothing to the total mass of the quadcopter

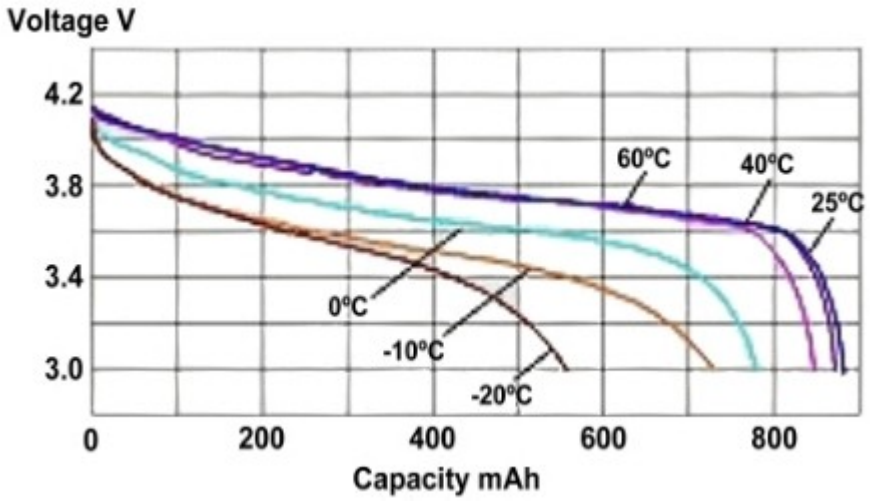
4.9 USAGE TEMPERATURES

Below is a list of the most critical components and their usage temperature ranges.

Component	Operating temperature range [°C]
LiPo battery	-20 to +60
LSM9DS1TR	-40 to +85
LPS25HBTR	-30 to +105
ATMega 32u4	-40 to +85

Table 4: Components' usage temperature ranges

LiPo battery can perform at more limited temperature range than the other components and its performance is affected the most by the temperature. Cold temperature decreases discharge rate and too warm temperature can decrease its lifetime or even overheat the battery. The best operating temperature for a LiPo battery is room temperature. Following chart shows typical LiPo battery discharging at different temperatures:



Drawing 7: 850 mAh LiPo battery discharge at different temperatures

The colder the temperature is, the faster the LiPo battery's voltage output drops and all of its capacity is not discharged. At 25 celsius degrees the battery fully discharges and voltage drop is slower than at high or low temperatures. Nonoptimal temperatures decrese flight time and thrust generated by motors.

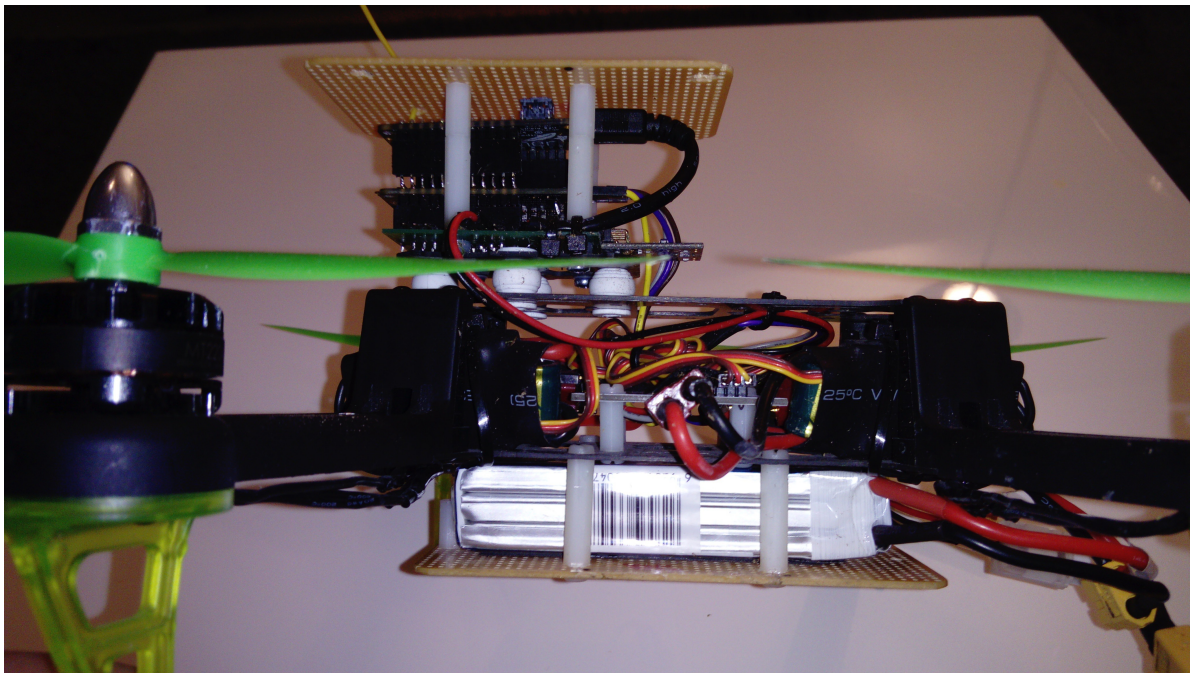
http://batteryuniversity.com/learn/article/discharging_at_high_and_low_temperatures

4.10 FRAME

All of the quadcopter's components are assembled on a frame. There are multiple sizes of frames for different types of quadcopters. Racing quadcopters have a small frame, small propellers, high motor rpm and fast maneuverability.

quadcopters that are used for photo shooting and recording videos usually have a larger frame, more mass, bigger propellers and lower motor rpm with higher torque. Generally they have slower maneuverability but can lift more mass.

Spendix Q250S racing size frame is used in this project. Its best feature is a board with rubber feet that can be used to install microcontroller and sensors on the frame. Rubber feet help to reduce oscillation of the frame caused by motors. The frame has also enough space for power distribution board and electronic speed controllers, legs that support the quadcopter when it is on ground and various holes on its carbon fiber frame which allows to install different components on the frame. Battery slot and circuit board are installed on the frame with plastic spacers as demonstrated by the following image.



Drawing 8: Drone's components from side view

It is optimal to install components so that their mass is distributed evenly on the frame. However it was not possible to install the board carrying the microcontoller and sensors at the center point of the frame, but fortunately their mass is relatively small compared to the mass of battery, electronic speed controllers and power distribution board. A properly implemented flight controller should be able to correct its orientation regardless of slightly uneven distribution of mass.

5 FLIGHT CONTROLLER

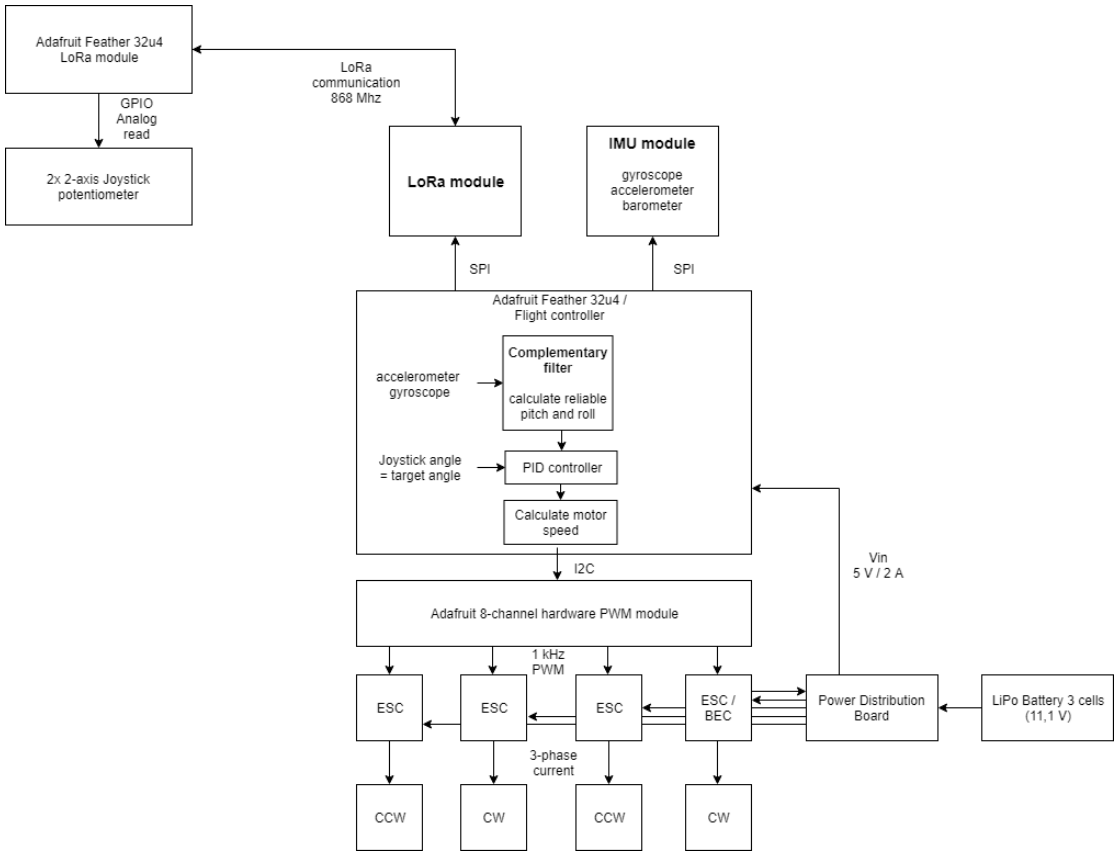
- Libraries used

All libraries are open source and some of them have been modified to satisfy the needs of the quadcopter.

- Arduino PID-controller
- RadioHead for LoRa communication
- Arduino basic libraries such as Servo library

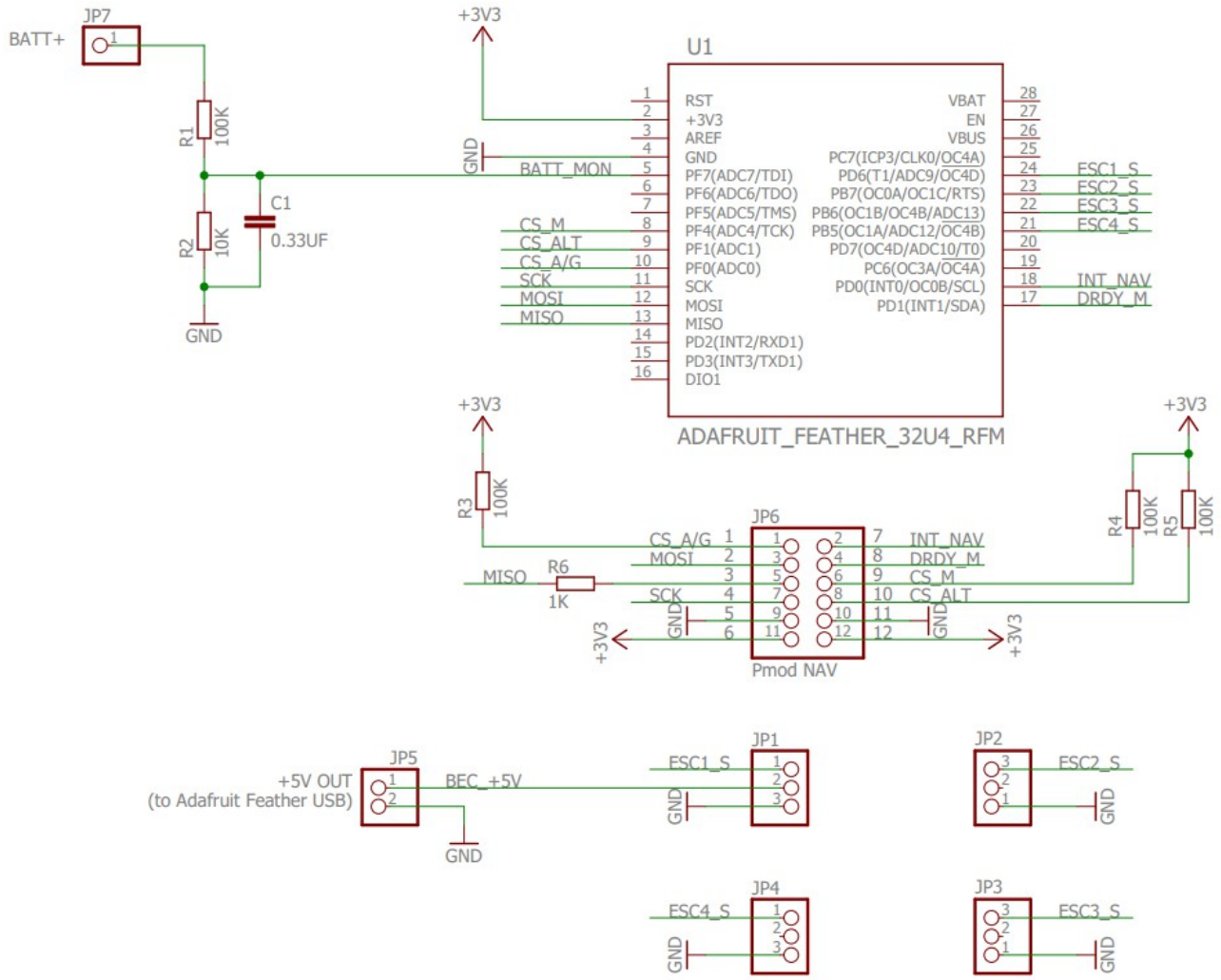
5.1 OVERVIEW

Flight controller is responsible for calculating current quadcopter orientation using accelerometer, gyrometer and magnetometer, receive target orientation from the user and calculate control orientation using PID controller. Control orientation will be used to calculate speed for each motor. Flight controller can support different flight modes such as acro and self-level modes. Self-level mode is implented in the thesis. It is harder to implement, but easier to fly. Self-level mode allows multiple future improvements, such as using GPS signal to locate the quadcopter and fly to a given location. It is also better option due to limitations of LoRa communication.



Drawing 9: Flight controller and joystick sequence diagram

5.2 FLIGHT CONTROLLER CIRCUIT BOARD



Drawing 10: Flight controller circuit board

Microcontroller, sensors and hardware module are connected together with custom made circuit board. Supply voltage comes from electronic speed controller's battery elimination circuit to JP1. JP1-4 are connected to ESC's PWM signal inputs.

Microcontroller communicates with IMU using SPI. This requires wiring MOSI, MISO, SCK and chip select pins to microcontroller. PmodNav has 3 sensors: accelerometer, gyroscope and magnetometer and each of them needs a chip select signal. Other connected pins are for receiving interrupts that signal data ready events. Microcontroller and PmodNav are connected to same ground and supply voltage. There are pull-up resistors connected to chip select signals, because chip select is zero active and the pin can be in floating state.

JP7 can be used to read battery voltage level, but it is not in use because software has not been implented for this feature. It has voltage division to lower the incoming 11.1 V to match microcontroller's 5 v logic level.

5.3 SOFTWARE ARCHITECTURE

Flight controller and LoRa joystick are two separate projects that both use communication and joystick components. Flight controller also uses IMU and motor controller components to perform all the required tasks.

5.3.1 Flight controller

Flight controller is the main component for the quadcopter implementation. It contains main function, so called flight controller loop, that will execute flight controller operations the whole run time. Each loop fresh sensor measurements are read, communication component is checked if there are any received packets and the packet is then parsed into joystick state. Joystick state and sensor readings are given to motor controller as input.

Flight controller loops needs to be as fast as possible to provide accurate approximations of the orientation, provide a steady PWM signal for the motors and read joystick input to provide a smooth and fast control of the quadcopter. ATMega32u4 has a 8 Mhz clock, configurable by fuses as 16 Mhz, by default uses clock divider 8 and runs on 1 Mhz system clock. This is quite low value, but for the time has provided control good enough. Clock divider can be changed to 1, resulting into 8 Mhz system clock which would allow 8 times faster performance for the flight controller. However this would require some study what libraries the change would affect. Arduino millis function values would be different, maybe Servo library that uses timers, etc...

5.3.2 LoRa joystick

LoRa joystick reads 3-axis potentiometer readings and uses joystick class to convert them into throttle, pitch, roll, yaw and press button commands 1 and 2. A LoRa packet is constructed from this data and it is sent using the communication component. Receiver will parse the payload into joystick state.

5.3.3 Communication

Flight controller and joystick use communication class to perform LoRa communication. Communication class handles receiving and sending packets as well as validating them.

5.3.4 Joystick

Joystick class represents joystick state and contains functions for

5.3.5 IMU

IMU class handles configuring sensors and reading sensor measurements from PmodNav using SPI (Serial Peripheral Interface).

5.3.6 Motor controller

Motor controller receives sensor readings and user commands as input. It uses complementary filter to calculate accurate estimation of quadcopter's orientation. Complementary filter output and user input are passed to PID controller as current orientation and target orientation. PID controller calculates control orientation that is used to calculate speed for all four motors.

Motors are controlled with pulse width modulation (PWM), which is software generated using timer interrupt. ATMega32u4 provides hardware PWM, but it is not capable of generating signal waveform required by the electric speed controllers (ESC).

5.4 SERIAL PERIPHERAL INTERFACE

Both peripheral devices, PmodNav sensor and sx1272 LoRa module, are communicated with SPI (Serial Peripheral Interface). Sensor has 3 chip select lines for accel/gyro, magnetometer and pressure. LoRa module requires 4th chip select line.
Device is active for SPI communication in when chip select is low. Only one device can be communicated at a time.

5.5 INTERRUPT SERVICES

The quadcopter has 2 interrupt services. One for receiving LoRa transmit/receive done and another for software PWM timer (more in PWM). IMU provides also different interrupts, but they are not used because flight controller loops runs a little slower than the sensor provides a new value, thus getting a fresh value on every read.
A specific interrupt order is important in the flight controller, because software PWM is used. On every timer tick motor PWM signal is updated and this needs to be consistent and fast to provide a proper signal so that motor rpm is smooth. This makes the timer highest priority and it needs to be able to interrupt lower priority interrupts, that is handling a received LoRa packet. LoRa packet handling is safe to interrupt. It is doing multiple SPI writes and reads and those are handled by hardware one byte at a time.
When sensor is being read via SPI, LoRa receive packet interrupt needs to be disabled, because both of them use SPI. Arduino beginTransaction function does not disable external interrupts, which LoRa receive is, and disabling the interrupt needs to be done by writing directly to processor's register EIMSK (External Interrupt Service Mask). When it was not disabled, the LoRa interrupt would start a new SPI transaction and the SPI hangs because sensor SPI communication was ongoing. SPI function will not return and only PWM timer interrupt is being handled providing the latest given PWM signal to the motors. At worst case this would be a high motor rpm and the quadcopter would fly uncontrollably away.

5.6 CALCULATING QUADCOPTER ORIENTATION

5.6.1 Complementary filter

Calculating current orientation of the quadcopter is important when implementing assisted flight mode. This is done by using accelerometer, gyroscope and magnetometer. Accelerometer measures linear acceleration on 3 axes. Gravity is is always visible on accelerometer. Using trigonometry on vectors results into orientation of the quadcopter, but accelerometer is sensitive for noise. The system is under a lot of resonation by motors, wind, etc.

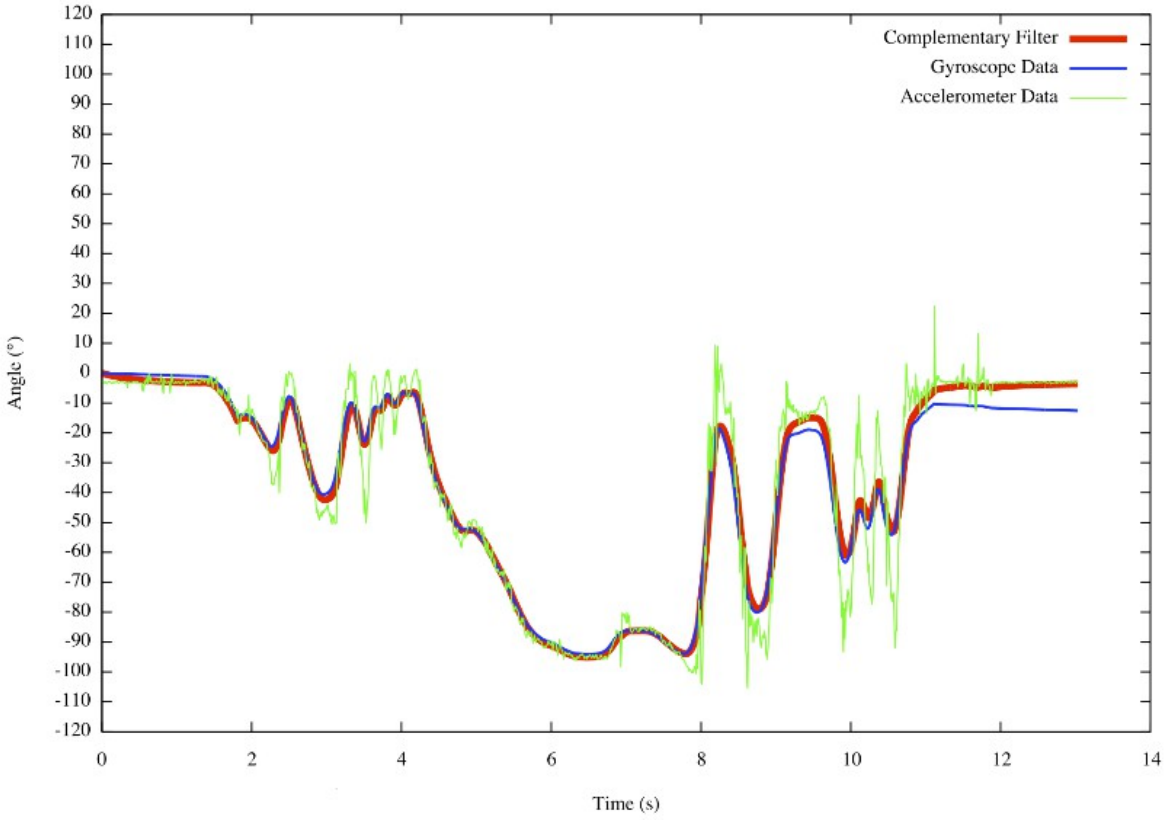
Gyroscope otherwise measures angular velocity on 3 axes, therefore quadcopter must be in angular motion to produce values on gyroscope. Gravity will not be visible on gyroscope and noise won't be visible. However gyroscope value drifts. To get the current orientation of the quadcopter is done by integrating angular velocity measured by the gyroscope. Due to inaccuracies such as floating point arithmetic, ODR of the sensor and speed of the flight controller loop will limit how often and accurately the value is read, as well as the inaccuracies of the gyroscope itself the value will start to drift. Meaning that when quadcopter is rotated around and brought back to the exact same orientation as in the beginning, the gyroscope value will be offset a little bit. The longer the time, the greater the drift value. Accelerometer does not have this problem, but as said, it is vaikutuksenalainen for a noisy system.

To use good capabilities of the both sensors, complementary filter is used to combine those values to as accurate approximation of the orientation of the quadcopter with a good performance. Complementary filter integrates gyroscope value and uses X amount of it's value, in the thesis 95%, and if the accelerometer reading is in a reasonable magnitude, we use that value in a trigonometric function to calculate orientation of the quadcopter and use 5% of that value. This compensates gyroscope drift and accelerometer noisiness.

5.6.2 PID-controller

In addition to calculating orientation of the quadcopter, it is required to control the speed of the motors according to calculated orientation and the target orientation given by user. These parameters are passed into a PID controller, where three configured terms are used to output a control orientation for the quadcopter. These terms are porpotional (P), integral (I) and derivative (D).
Porpotional is the subtraction of current orientation and target orientation. This simply provides the error margin between those values. Using only this term would cause very oscillating system, where the quadcopter would bounce around the target orientation never reaching it in a smooth manner.
Intergral term would slowly increase when the quadcopter is not in its target orientation. This will fix the quadcopter drifting away in a windy environment. It does not affect the quadcopters orientation a lot unless it has been in a wrong angle for longer time.

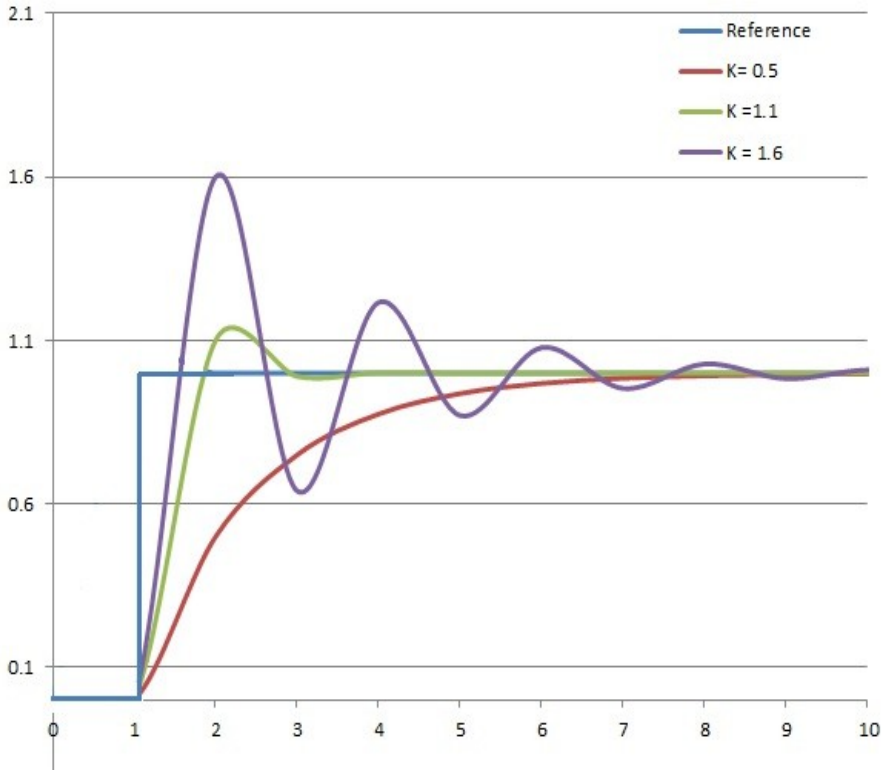
Derivative term will dampen the aggressive corrections made by porpotional term. Without dampening, noise left by the complementary filter would affect porpotional term too much. Derivative term will bring too big corrections down.



Drawing 11: Complementary filter

Porpotional term controls error margin
Integral term controls cumulative error
Derivative term estimates trend of error margin and dampens it

$$u(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e(t')\,dt' + K_\mathrm{d} \frac{de(t)}{dt},$$



Drawing 12: PID controller configured with different porpotional term values

5.6.3 PID-controller tuning

Tuning should be started with porpotional term. Initially set P value to 0 and increase until the quadcopter can be controlled indoors, but the quadcopter makes steady small oscillation natural to the P term. Then increase the I term until the oscillation gets smoother and the quadcopter does not start to drift. Increase P term until the drift stops and start increasing I again until drift is minimal. Then D term until there is no drift. ????

5.7 CONTROLLING QUADCOPTER ORIENTATION

5.7.1 Calculating motor rpm

PID controller outputs the control pitch, roll and yaw (maybe also throttle?) for the quadcopter. This output will be used in an algorithm to calculate the actual PWM duty cycle for the ESCs.
The algorithm is roughly this for a X model quadcopter:

```
CW front motor = targetThrottle - controlPitch - controlRoll - Yaw
CCW front motor = targetThrottle - controlPitch + controlRoll + Yaw
CCW back motor = targetThrottle + controlPitch + controlRoll + Yaw
CW back motor = targetThrottle + controlPitch - controlRoll - yaw
```

This means that when the quadcopter is maneuvered to fly forwards, the back motors will be given more throttle than the front motors. When maneuvering to

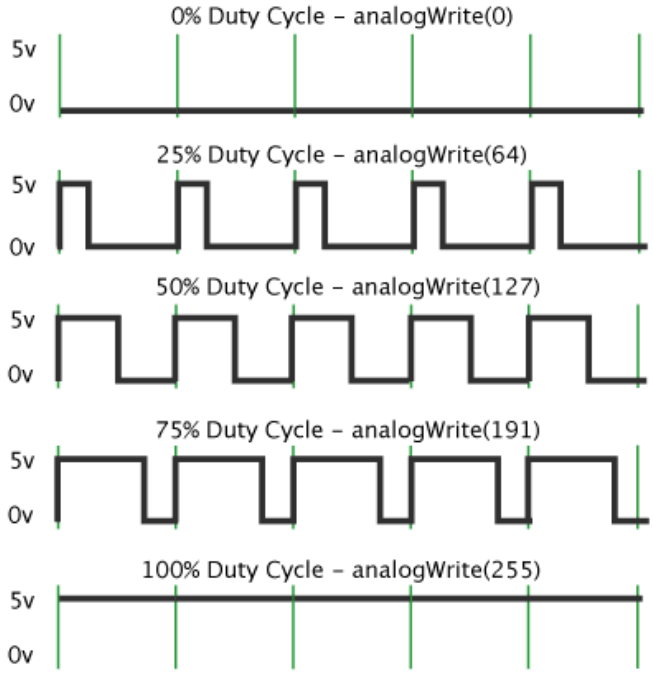
left, right front and back motor will have higher RPM than the left motors. To yaw right, right top and left back motors are given more throttle.
Maximum value for motor speed is 255 (because joystick input is in range of 0...255). Giving only full throttle would result to 255 for each motor, considering that the PID controller does not need to stabilize the orientation of the quadcopter. This is given as percentage multiplier to PWM signal (180 * motor speed %). The range provides enough accuracy: (-30...30 angle) -> 60 / 255 = 0,23 degree steps. And 255 steps for the throttle. This comes pretty close to the joystick accuracy also.
The motor control value can go over the limit of 255 when giving close to full throttle and tilting the quadcopter or giving close to zero throttle and tilting the quadcopter. Therefore the value needs to be clamped inside the range of 0...255. Highest over or under limit value is checked, and it is summed or subtracted from all of the motor values providing the same difference between motor duty cycles, but in the given range.
These motor values will be divided by 255, giving the value in percentages. The servo library duty cycle parameter will be multiplied by this (180 * (0...1)) to give proper PWM signal for the motors.

Also a + model quadcopter can be implemented, where the quadcopter will have one motor at front, one at back, and one at each side.
The X model copter has two front motors and two back motors. This is the most used model in commercial flight controllers, but they might have an option to change between these modes. What's the main difference between them??
What should the difference between motor RPMs be???

5.7.2 Pulse width modulation

Pulse Width Modulation (PWM) is a common way to control servo motors and ESCs. Quadcopter ESCs implement custom PWM firmware and there are different signals to command the ESC. BLHeli ESCs used in the thesis can be controlled with two types of PWM signal.

First is the usual servo motor PWM signal at 50...500 Hz frequency where period is 2...20 ms. Usually control signal would be from 1 to 2 ms, but in BLHeli ESC it is from 1.2 to 1.8 ms. For this reason frequency cannot be higher that 500 Hz, because then period would be shorter than 2 ms. When throttle is 0 %, signal is set high for 1.2 ms and for 100 % throttle signal is set to 1.8 ms. This provides safety in situations where signal would be erroneous due to problems in the program or hardware. A digital port in error state would not be able to generate this specific control signal, but rather being low or high all the time.



Drawing 13: PWM signal at different duty cycles
https://www.arduino.cc/en/Tutorial/PWM

BLHeli can also be used with 1 kHz or higher frequency. In this case duty cycle directly represents motor speed. Control signal is set high from 0 to 100 % of the period. For example setting duty cycle to 25 % of the period, the motor speed would be 25 %. If the frequency is 1 kHz, it is possible to change motor speed every millisecond. That is 50 times faster than with 50 Hz signal. However this is prone to error states where digital port would be high all the time and command motor speed to 100 %. To eliminate this problem custom hardware needs to be implemented.

1 kHz PWM signal was chosen for the thesis because of speed and more importantly that hardware PWM modules used were not able to generate 50 Hz signal with 1 to 2 ms control signal as well. This would have been possible with software timer interrupts but it requires more performance from the microcontroller and more complex code to configure interrupt priority.

This needs to be done in software using timer interrupts. These are executed every x us, variating because... Handling these interrupts is the priority task of the flight controller, so that motors are provided with accurate PWM signal. During development there were other interrupts blocking the PWM timer interrupt and it resulted into erroneous signal, motors sometimes accelerating and deaccelerating in fast manner, making a whizzing sound and unsteady control.

5.8 FLIGHT SAFETY

Flight controller provides functionalities to safely start the flight from inactive state and land when the LoRa communication between the quadcopter and joystick fails.
To start the flight user must slide throttle to zero so that the quadcopter does not suddenly start flying when it is powered or has landed. If no LoRa packets have been received for 400 ms, the quadcopter will stabilize itself and give low throttle for the motors. In this mode the quadcopter will start slowly descending to ground. If communication restores, the user gains control of the quadcopter again. After x seconds of no packets received the motors will be turned off and flight will be inactivated. If communication restores, user needs to slide the throttle to zero to activate the flight again.
If the quadcopter code errors, no valid PWM signal will be generated for the ESC and motors will shutdown.
TODO: Every motor update a timestamp is updated and checked in an interrupt service. If the last update was executed over 400 ms ago, the flight will be inactivated and the motors shutdown. This prevents errors where some procedure, like SPI communication, hangs up but the actual code does not error. Without this interrupt service the PWM signal is still generated because timer interrupts are running and the quadcopter will fly uncontrolled.

5.8.1 Flight inactive

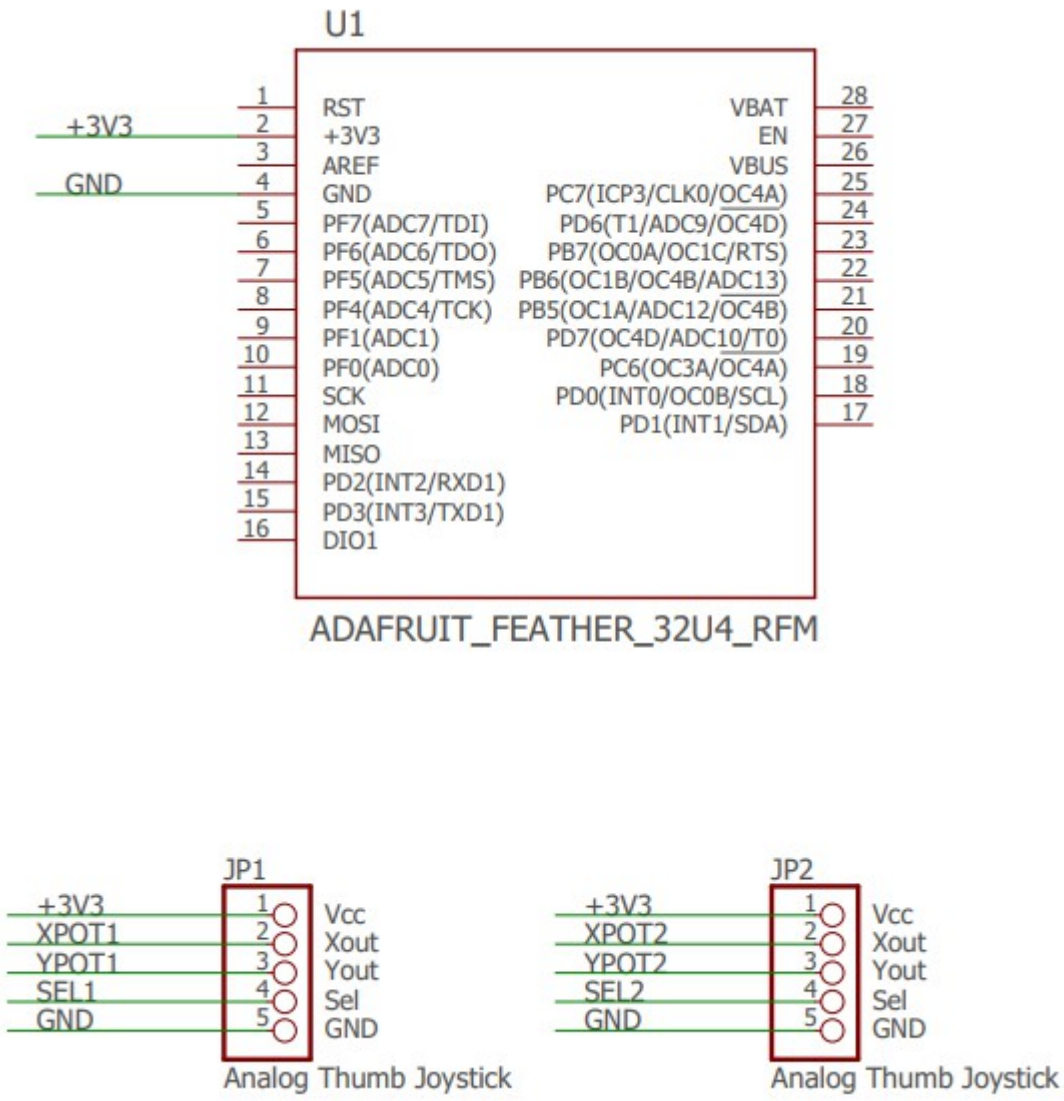
Flight actives when throttle is 0. This provides a safe way to start the flight ensuring that throttle can be slowly increased. Flight is inactivated by pressing right joystick for 1 second (and letting throttle go from bottom to center). When flight is inactive quadcopter's PID values can be configured with right joystick. Right press will cycle the term in order porpotional (P), integral (I), derivative (D). Up press will increase the term value by X and down press will decrease the term value by X. Holding joystick in left position for 1 second will set the terms to default values. Giving one of the commands will blink a red led on the quadcopter once the right stick brought back to center position.

6 JOYSTICK

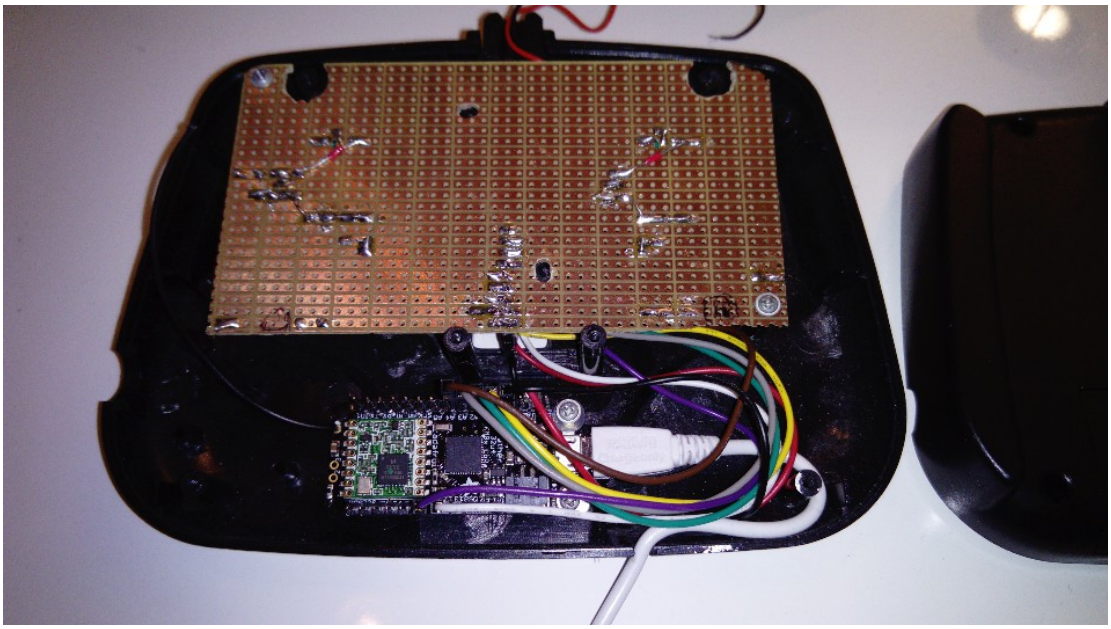
6.1 COMPONENT LIST

2x 3 axis joysticks from an old PS1 controller
Reikälevy and copper wires
Arduino Feather 32u4
Resistors
An old RC car controller case modified

6.2 JOYSTICK CIRCUIT BOARD



Drawing 14: Joystick circuit board



Drawing 15: Joystick viewed from inside

6.3 JOYSTICK FUNCTIONALITY

Each loop reads joystick values using ADC for X and Y axis of the both joysticks and digital comparison for Z axes. These values are converted from int16 to byte, because the range is enough, it uses less bytes in the payload. Center position integer values is 125, full throttle 255 and no throttle 0. This loop is lightweight and only the LoRa communication sets limit for the packet send ratio, which is roughly 10 (try this with packets per 10 s print, so that Serial does not slow down) packets per second.
These bytes are sent as payload to the quadcopter using LoRa. The LoRa communication is explained in section ... and packet integrity and security in ...
Left joystick X axis represents yaw and Y axis throttle. Right joystick X axis represents roll and Y axis pitch. Explain the maneuvers here or on some other section.
Z axes are used for commands such as inactivate flight mode.

6.4 JOYSTICK COMMANDS

Joystick provides several commands to set PID-controller values, user input dampening, setting the quadcopter in flight and inactive mode, and resetting all the settings to default values. The quadcopter indicates which command is selected and how it is changed by blinking Feather's default led.
[1][2][3][2]

[4]

Command	Function
Turn right joystick right	Rotate between P, I and D terms and user input dampening. Led blinks n times for each parameter: 1: P 2. I 3: D 4: dampening
Turn right joystick up	Increase selected parameter value. Led blinks twice.
Turn right joystick down	Decrease selected parameter value. Led blinks once.
Hold right joystick left for 1 second	Reset all settings to default. Led blinks for 2 seconds.
Press right joystick and hold throttle (left joystick) down	Inactivate flight mode
Hold left joystick down	Activate flight mode

Table 5: Joystick commands

7 COMMUNICATION

7.1 LORA PROTOCOL

LoRa is Long-Range Low Power radio communication protocol that is made mostly for industrial use for IoT (Internet of Things) implementations. It provides long range communication with low power (Semtech sx1272 around 120 mA during sending). It was chosen for this thesis, because we had a course at VAMK where we used LoRa and because it has simple libraries and payload to send all the required data to the quadcopter, and if needed also from quadcopter back to the joystick. For example sending battery state to the joystick. Also the Adafruit Feather devices have preinstalled LoRa module that is easy to configure for use. It uses 868 Mhz which is allowed hobbyist frequency in Europe.

7.2 JOYSTICK AND QUADCOPTER COMMUNICATION

RadioHead library is used to communicate with the LoRa module. It handles initialization of the registers, switching between LoRa modes, receiving interrupts when a packet is received and parsing the LoRa packet and providing the payload. Adafruit Feather 32u4 has interrupt pin 6 connected to the LoRa module interrupt service. It is used for transmit and receive done interrupts. Every flight controller loop a flag status is checked. When RadioHead has received and parsed a LoRa packet, it sets the flag to true and the payload can be accessed by using library function recv.
The payload will be copied to a buffer given as a parameter to the function. In this case, the payload buffer needs to be 7 bytes.

7.3 PACKET VALIDITY AND SECURITY

Since the quadcopter might receive any LoRa packet, not only the ones sent from joystick, the packet needs to be validated to make sure that it comes from the joystick, contains proper data and is not corrupted during transmission. Packets are sent continuously as fast as possible, quadcopter receiving at least 10 packets per second, and at the same time it is calculating its orientation, controlling PWM signal interrupts and other routines. For this reason only lightweight algorithms are used. Packet contains 6 bytes of joystick data (pitch, roll, yaw, throttle and 2 buttons) and the last byte is reserved for CRC (cross cyclic redundancy). CRC is calculated from the 6 data bytes.This gives some data integrity if the payload is corrupted (LoRa probably does this on its own??) and also makes identification that the packet is not any random packet sent by another LoRa device. These 7 bytes are XORed using a 3 byte XOR key to add some more data integrity and validation and making the intepretion of the packet slighty harder for receiver that it does not belong to.
LoRa packets are often encrypted with SHA256 or other strong encryption mechanisms, but to provide fast packet handling they are minimalistic in this project.

7.4 LORA PROTOCOL IN QUADCOPTER COMMUNICATION

- Slower than the usual communication method used in quadcopters (explain what it is)
- LoRa network can be used to control more self-sustaining quadcopters that use GPS locations as control input. Command multiple quadcopters etc.

8 DRONE FLYING AND FINNISH LAW

Drone is an unmanned flying vehicle that is meant for hobby or sport usage. There are various details in Finnish law that concerns flying a drone. Aviation law is available at Finlex and Trafi's Q&A page answers the most important questions for a private person.

https://www.finlex.fi/fi/laki/alkup/2014/20140864 864/2014,159 §
https://www.trafi.fi/tietopalvelut/usein_kysyttya/ilmailu_-_miehittamattomat_ilma-alukset_ja_lennokit

- Flying a drone is not age restricted.
- The person controlling the drone needs to have constant line of sight to the drone.
- A drone can be flied in FPV (first person view) if there is another person that can observe the airspace and has constant line of sight to the drone.
- A drone cannot be flied at altitudes higher than 150 meters.
- Weather and lightness needs to be good enought to spot other flying vehicles and evade them with using bare eye contact without using any equipment
- A drone can be flied indoors if the owner of the building accepts it.
- A drone needs to evade all manned flying vehicles.
- It is against law to fly a drone in locations, such as an airport, where it will disturb manned. flying vehicles or otherwise cause any dangerous situation.
- An autopilot can be used, if you can at any time take the control of the drone and evade people and other flying vehicles.
- Flying a drone on your own property can still violate someone's privacy. In this case they can contanct Trafi to reclamate.
- A drone does not need to be registered.
- Lentäjän vastuu määräytyy vahingonkorvauslain (412/1974) perusteella.

9 CURRENT STATE OF THE PROJECT

9.1 CURRENT IMPLEMENTATION

9.2 REQUIRED IMPROVEMENTS

Most of the required functionality is implemented and working, but there are some aspects that should be improved to stablize the flight. Currently the drone is very unstable and easily flips on its back.

9.2.1 Flight controller loop frequency

Many commercial flight controllers have a flight controller loop running at 1 kHz frequency. It can be even as fast as 8, 16 or 32 kHz. The frequency of flight controller loop implemented in thesis has been around 180...400 Hz during different development stages. A small racing size drone requires a fast executing flight controller loop, because they have less mass, higher motor rpm and fast maneuvers. This seems to be one of the limitations in the current implementation, because during flight it can be seen that the flight controller is trying to fix the orientation but does not do it fast enough or does fix the orientation too much, flipping on its back.

There should be various optimisations that improve execution time of the flight controller loop. For example external hardware PWM module should not be used, because SPI communication to it takes few milliseconds each loop and that has a huge effect on performance. This requires changes in the circuit board and PWM signal generation.

Minor improvement would be using quaternions instead of euler angles for calucating drone's orientation, because quaternion's are slightly faster to compute. However this alone would not affect the performance a lot. There are likely some other bottleneck's that should also be addressed.

9.2.2 Yaw calculation

Currentyle the flight controller does not use a magnetometer to compensate gyroscope's drift when calculating yaw. This is due to configuration and calibration problems with the magnetometer and therefore it does not provide reliable measurements. The drone tends to yaw to right and this affects the flight when gyroscope drift grows.

9.2.3 PID controller tuning

After the above major errors have been fixed, the drone requires PID controller tuning to stabilize the flight correctly. Currently PID controller tuning affects flight performance of the drone, but other problems in the flight controller prevent tuning and testing PID controller properly.

10 CLOSURE

During the project I learned a lot of detailed information about drones, embedded systems programming and hardware design. Implementing a flight controller is a challening task because there are various things at hardware and software level that need to work together in a fast changing system and a demanding environment. I will continue my drone project as a free time hobby - improving the flight controller and implementing fancy features like FPV camera and battery monitoring.

I would like to thank VAMK, University of Applied Sciences for supporting component purchases and allowing me to do this interesting thesis project, Jani Ahvonen who is my thesis supervisor and especially my father who helped me a lot with hardware design.

REFERENCES

0: Aircraft principal axes, ,
0: Quadcopter flight configurations, http://ardupilot.org/copter/docs/connect-escs-and-motors.html
0: Brushless DC motor, http://www.thinkrc.com/faq/brushless-motors.php
0: 3-phase brushless DC motor steps, https://www.digikey.com/en/articles/techzone/2013/mar/an-introduction-to-brushless-dc-motor-control
0: Propeller, https://www.slideshare.net/tomcpaulus/intro-to-multicopters