Gradu Amaierako Lana / Trabajo Fin de Grado
Ingeniaritza Elektronikako Gradua / Grado en Ingeniería Electrónica

Study and characterization of the electronic and control systems involved in the flight of a quadrotor.

Exploring sensor aplications

Egilea/Autor/a:
Miguel Astrain Etxezarreta
Zuzendaria/Director/a:
Mariano Ruiz Gonzalez

Leioa, 2016ko Otsailaren 24a /Leioa, 24 de Febrero de 2016

ACKNOWLEDGEMENTS

I wholeheartedly thank Professor Mariano Ruiz of the Escuela Técnica Superior de Ingenieria y Sistemas de Telecomunicación at Universidad Politécnica de Madrid (UPM). Prof. Ruiz and I did not know each other before this project and he offered all his help and time to develop this project.

I would also like to thank Prof. Josu Jugo García of the Facultad de Ciencia y Tecnología at Universidad del País Vasco (UPV) for his assistance on the duties of Codirector and his willingness to help.

Lastly, I would like to thank Prof. Alfredo Lorente Fillol and Sviatoslav Ibañez Fernandez of the Instituto de Fusión Nuclear at Universidad Politécnica de Madrid (UPM) for their expert guidance on radiation testing.

Miguel Astrain Etxezarreta

Disclaimer: This document was first drafted for UPM Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación but due to delays and administration problems it was transferred to Facultad de Ciencia y Tecnología at Universidad del País Vasco (UPV). Format and length has been accommodated as far as was possible.

Abstract:

This project entailed the design and setup of and outdoor quadrotor to be used as a platform to test possible sensor applications.

As the project advanced, focus shifted from descriptive characterization to a more practical approach.

The final build is a medium sized quadrotor for which a Geiger-Müller type sensor has been developed and integrated.

Resumen:

Este proyecto se centra en el desarrollo y la caracterización de un dron como plataforma para el desarrollo de aplicaciones con sensores.

El proyecto ha evolucionado desde su inicio, siendo al principio un proyecto para poder analizar los sistemas y algoritmos que conforman un dron.

Como resultado final se ha desarrollado un sensor Geiger-Müller que transmite sus lecturas a un ordenador por radio y se ha descrito detalladamente los pasos y procedimientos a seguir para el desarrollo de un sistema igual o similar.

Laburpena:

Proiektu honen bitartez droi bat garatu eta karakterizatu egin da, sentsoreen garapenerako plataforma moduan erabilita.

Proiektua garatzean lanaren hasierako izatea moldatuz joan da. Hasieran sistemen karakterizazio sakona genuen helburu, baina aplikazio praktikoago bat izan da gure helmuga.

Geiger-Müller motako sentsore bat garatu da, irrati bidez ordenagailu bati datuak bidaltzeko gai dena.

Contents

Acronyms:

UAV	Unmanned Aerial Vehicle
ARF	Almost Ready to Fly
PCB	Printed Circuit Board
GPS	Global Positioning System
ESC	Electronic Speed Controller
IMU	Inertial Measurement Unit
G-M tube	Geiger-Müller tube
DCM	Discontinuous Current Mode
Li-Po	Lithium Polymer
ADC	Analog-to-Digital Converter
PPM	Pulse Position Modulation
PCM	Pulse Code Modulation
PWM	Pulse Width Modulation
IDE	Integrated Development Environment

Introduction:

Robotics area has quickly developed throughout the whole century and drones are the very first representatives of this development. What really made drones stand out is how cheap they are, commonly converted into toys, we can find lots of different models in the market. Usually described by the number of motors they got, quadrotors are the most common form of drones nowadays. But drones are far from being toys, they are very capable and mobile platforms that we can use to navigate in other ways impossible to access areas.

Five years ago, in 2011, a terrible disaster occurred in the Fukushima region in Japan. This catastrophe was initiated by a tsunami that damaged the coolant equipment of the Fukushima Nuclear Plant. Afterwards, global concerns about nuclear energy were raised. Japan literally switched off all its nuclear power plants to infuse tranquillity. In the future we might be able to sustain our needs just by using renewable energy, but until then we are dependant of carbon and nuclear energy. [1]

These kind of tragic episodes may occur inevitably, although very rarely, and we have to be prepared to respond. Drones are capable of reaching the most inaccessible places faster, moreover, deploying a drone with the right sensor kit could be crucial for acquiring fast information. Or they could also do routine patrol tasks around hot zones, like nuclear plants, to make our daily life a bit safer.

Our objective here is to design and setup a quadrotor as a platform for the development and integration of sensors. In our case, a Geiger-Müller radiation detector has been developed and mounted onto this quadrotor but the work is also useful for any sensor interfacing with this setup.

This project was developed in parallel to a college year, as a degree final project at Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación of Universidad Politécnica de Madrid. As for that, time constraints where critical on the design considerations.

Keeping this in mind, we also need it to be as accessible as possible to modifications, so we went for the open source route. The controller we selected is a modified version of APM 2.6 made by the community effort of Arducopter "This is the full-featured, open-source multicopter UAV controller that won the Sparkfun

2013 and 2014 Autonomous Vehicle Competition (dominating with the top five spots). A team of developers from around the globe are constantly improving and refining the performance and capabilities of Copter". [2]

This controller is perfect for our development because we can easily access all the information concerning how their hardware and software works and has a huge community backing it up at diydrones.com. This project is also very interesting from an engineering and telecommunications standpoint as we define, describe and implement every component in our way to building a quadrotor.

This project, with all the technical and software considerations mentioned above, has given rise to the final setup that can be seen in the following figure (Fig 1).

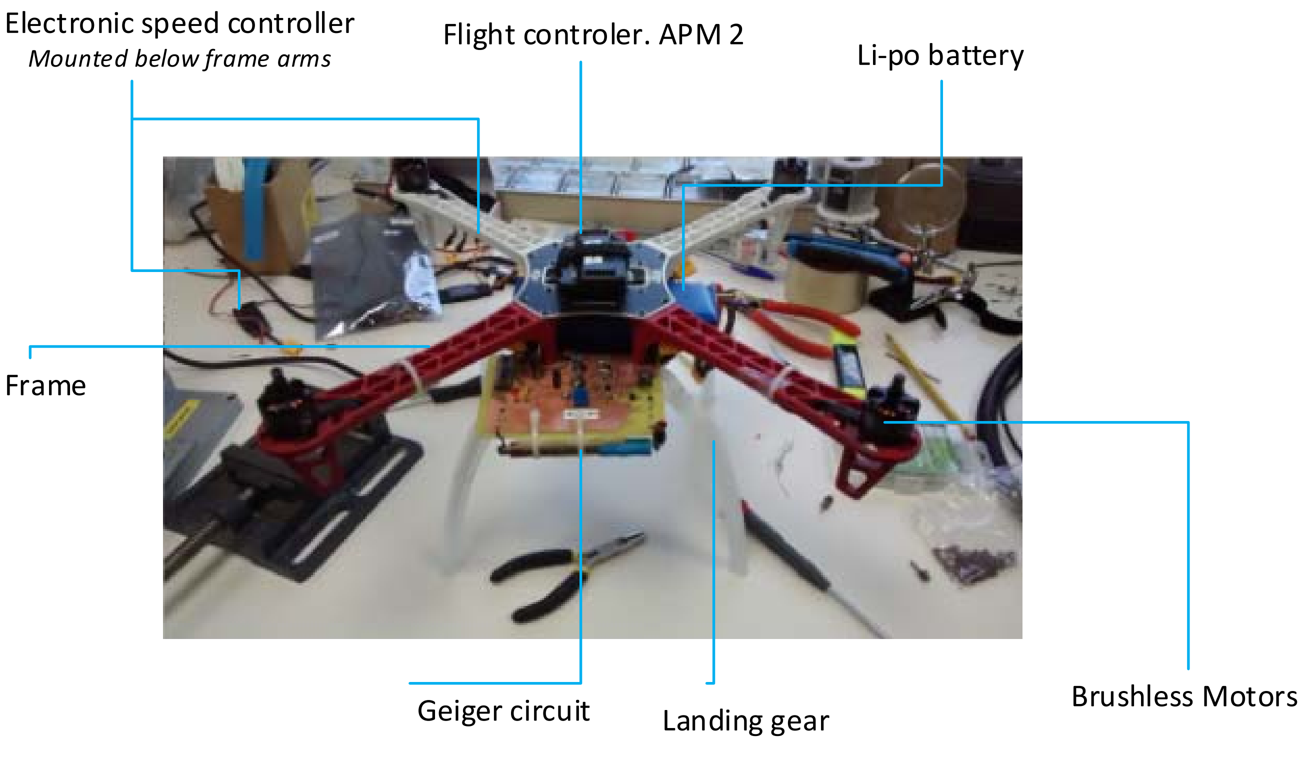


Fig 1: The most important components of the drone showcased over the work table. Each relevant component is highlighted.

Chapter 1: Building a Quadrotor

1.1 Design considerations

There are many different setups for quad rotors, therefore, we can specialize the build to our needs. We will start our task by presenting our requirements.

In general you would consider things like flight time, payload (closely related to the first one) and camera/video options. But as we are developing a sensor platform, the payload is not going to be an issue, as the majority of the sensors and the associated circuitry are not very heavy.

The most important design considerations are:

- Low price
- High fly time
- Space to attach circuit boards
- Land Station interfacing
- Data storage
- Easy setup

1.2 Component description and characterization

1.2.1 ARF kit

When we purchased the frame, the best deal was what is called an almost ready to fly package (ARF).

What is interesting about these packages is that you get almost everything you need; it includes motors, electronic speed controllers, frame and propellers. All of the parts have been tested on a drone and work well together, therefore the first time you are planning on building a drone those kind of packages are very helpful. We need to remember that our goal here is, not only to fly, but to develop a sensor to test on the drone.

As for the system recommended specifications we should aim for a total payload of around 1200g. The choices will be explained more extensively on the sections ahead.

1.2.1.1 Frame

The first thing to consider is the size of the quad rotor we want to build, as it depends a lot on the application we want to give it.

The frame choice deviated from the open source mindset for a few reasons. Firstly, lack drone piloting experience, therefore a robust frame will help us prevent severe damage during development. Secondly, we will be developing our own sensor boards for this project so we can design it to fit on a frame we already have. For this reason we went for a commercial drone frame which has been tested by a huge community and has proven it is very tough.

1.2.1.2 Electronic Speed Controllers (ESCs)

Drones use brushless motors. To control such motors we need one controller per motor. These controllers are known as Electronic Speed Controllers (ESC). A three-phase current is delivered via 3 wires to the motors, creating a voltage differential across the motor terminals and making them spin.

The flight controller generates a pulse-width modulation (PWM) signal. The conversion is made by the ESC's, which draw battery power and supply the motors.

The DJI 15A OPTO ESC will take a 5 V PWM signal as input and translate it to 0% to 100% of the thrust. As we are using this particular ESC, calibration is not needed but should be checked in other cases.

The specifications for this ESC’s (see Fig 2) are listed in Table 1:

Table 1: Basic specifications of the ESC used

Current	15A
Signal Frequency	50Hz to 450Hz
Voltage	11.1 to 14.8 V
Battery	3S or 4S

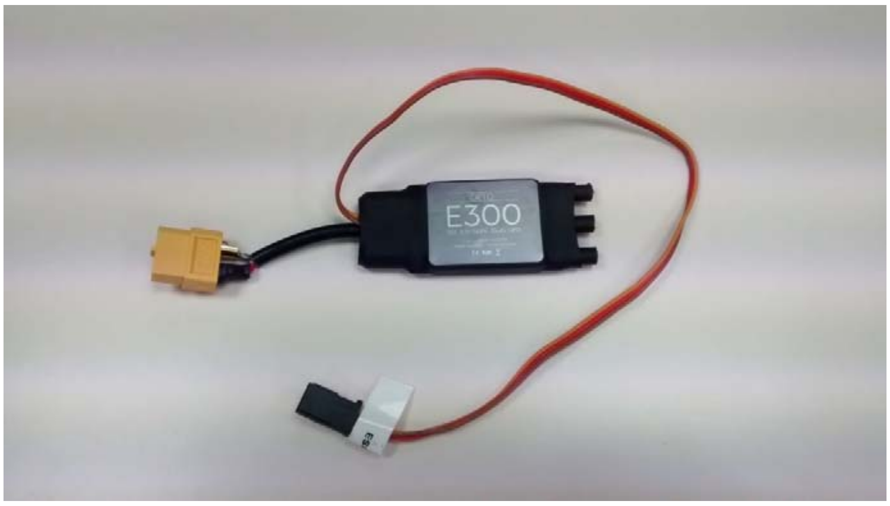


Fig 2: E300 electronic speed controller. Battery feeds through XT-60 connectors (Yellow). The 3 pin connector drives the PWM signal from the flight controller.

1.2.1.3 Motors

The kit specifies its E300 propulsion system with some very vague specifications, but does point out these are 2212 motors (Fig 3) manufactured by them so we will be using those specs instead(Table 2) :

Table 2: 2212 Motor specifications

Dimension	28X24 mm
Rating	920 kV
Shaft	8.0 mm
Weight	56 gr
Standard Current	15-25A Max Current : 30A
Recommended propeller	Li-Poly (11.1V) : 10x4.5 inch Li-Poly (14.7V) : 8X4.5 inch

The motor specifications differ slightly of those on the ESC because their max current is well beyond ESC specifications. A 3S battery will power these just fine, as the 15A ESC's are also over dimensioned but manufactured as 15A OPTO ESC's. So the kit will fly perfectly and will also be very resilient to mistakes. [3]

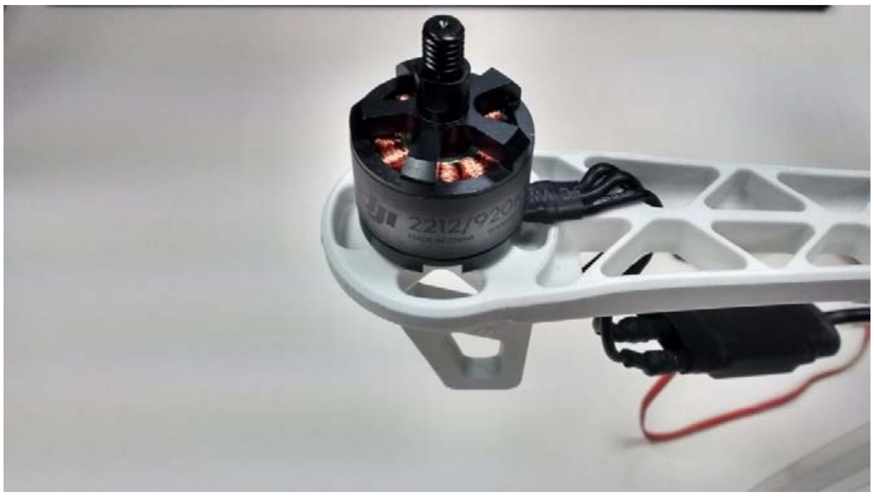


Fig 3: 2212 brushless motors connected by 3 bullet connectors to the ESC's. Four screws hold it to the frame (White). The ESC (under the frame arm) will be tied to the arm with bridles.

1.2.1.4 Propellers

Propellers are defined by two numbers, the first one is the diameter of the propeller and the second one specifies the curvature of the blades. It is near impossible to find propeller specs in millimetres, therefore, for convenience, we will refer to them in inches.

With more custom builds we would expect around10 inch propellers for 3S batteries and 8 inch for 4S, but instead the DJI propellers are 9.4 x 4.3 inch.

The DJI 9.4 inch propellers are auto-adjustable, meaning you do not need any other piece of equipment to hold them to the motors. The drawback is the cost because we will pay a bit more per propeller but will gain in reliability. Based on those 9.4 inch propellers there are other manufacturer's propellers (Fig 4) compatible with the auto-lock system for our quadrotor, in case DJI stops manufacturing them. [4]

Fig 4: Non DJI auto lock propellers.

1.2.2 Battery and Charger

Once the frame was chosen, and as the ESC and motors come with it, we can choose the battery for our quad rotor. All modern drones use Lithium-Polymer (Li-Po) batteries as their high output current capabilities are the required by the motors and they are very light weight compared to other technologies.

These batteries are made of cells. The cell of a Li-Po battery outputs high currents at a minimum voltage of 3.7 volts. Usually this voltage is not enough or is not used to supply the motors so the batteries come packed with some cells in series.

For our DJI F450 we could choose 3S or 4S packs, which means you got 3 or 4 Li-Po cells packed in series for minimum voltages of 11.1 V and 14.8 V respectively.

Smaller propellers need more speed to generate the same thrust, 3S batteries will give us a slower flight, but after reading about user feedback and how the frame behaves [5] we decided that a 3S Li-Po battery (Fig 5) with the DJI standard propellers will work fine. We could choose 4S packs in case we went for an acrobatic drone.

Table 3: Turnigy battery specifications:

Capacity(mAh)	5000
Config (s)	3
Discharge (c)	40

Weight (g)	443
Max Charge Rate (C)	5
Length-A(mm)	145
Height-B(mm)	50
Width-C(mm)	27
RC transmitter battery	
Capacity(mAh)	2200
Config (s)	3
Discharge (c)	1.5
Weight (g)	139g
Length-A(mm)	100
Height-B(mm)	33
Width-C(mm)	19

Table 4: Turnigy battery specifications.

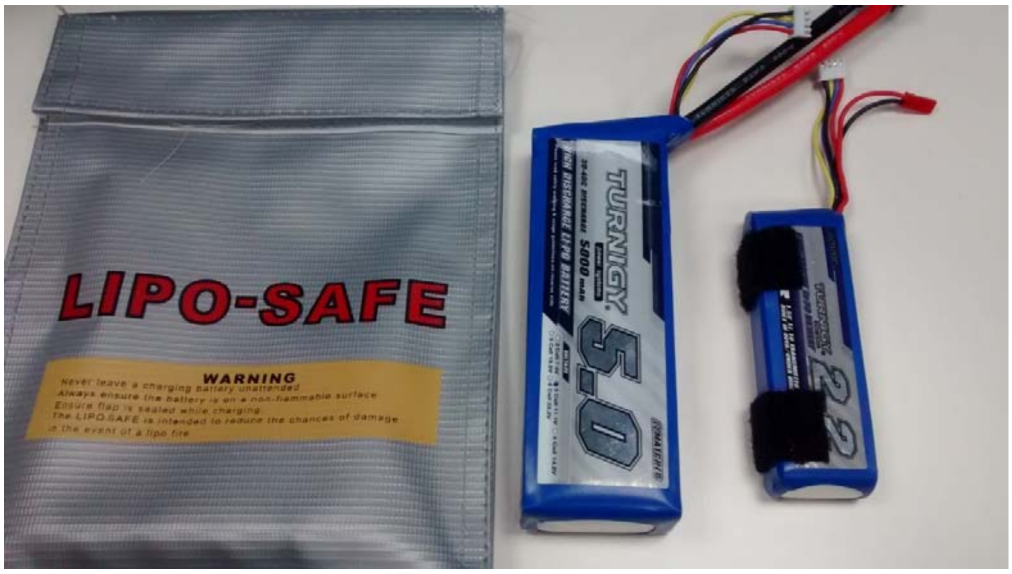

LIPO-SAFE
WARNING
Never leave a charging battery unattended
Always ensure the battery is on a non-flammable surface
Ensure flap is sealed while charging
the LIPO SAFE is intended to reduce the chances of damage in the event of a lipo fire
TURNIGY
5.0
5000 mAh
2.2

Fig 5: Left, fire resistant safe bag to store Li-Po batteries. Middle, 5000 mAh 3S battery pack for our quadrotor. Right, 2200 mAh 3S battery for the radio transmitter.

We usually refer to batteries by their capacity but when talking about drones the C rating of the battery is very important. Capacity for a battery is determined by how much current gives over an hour long discharge. By this definition a 5000 mAh battery would give us 5 amperes over an hour.

Unloading a battery faster than that will usually mean the total capacity is reduced as the efficiency of the battery drops with fast unloads. Some batteries might even be permanently damaged due to this.

For quadrotors the best battery technology is the Lithium Polymer, these batteries are able to output high currents in very little time, for example, to take off. Li-Po batteries also can have high C ratings which means they can unload that number times the nominal current. For example, our battery has a 40C rating and 5000 mAh capacity, which means it would be able to yield up to 200A without getting damaged.

As our ESCs, which will draw the most of the power out of the battery, can consume up to 15 A we are safe using a 5000 mAh 40C battery.

The battery choice was also determined by a big sale weekend, this battery was particularly economic, my guess here it is a bit too heavy for the common quadrotor user, but we had the space in our quad and high end performance is not a goal for us.

With the sale we could also get a very basic and simple Li-Po battery charger (Fig 6). The charge up time is of about 4 hours (1.2 Amps output current). When using Li-Po batteries you have to control their charge with caution, over charging or discharging could damage the battery or even set it ablaze.

Fig 6: Basic Li-Po charger. Can charge 1S, 2S and 3S battery packs.

1.2.3 Power Distribution

To draw power from the battery to the flight controller we need a regulator as the battery supplies over 11 volts and we need steady 5 V for the APM. We chose a very handy model from HobbyKing, the component manufacturer, which is not only a 5V power supply but also comes with current and voltage sensors (Table 5) so we can measure battery status (Fig 7).

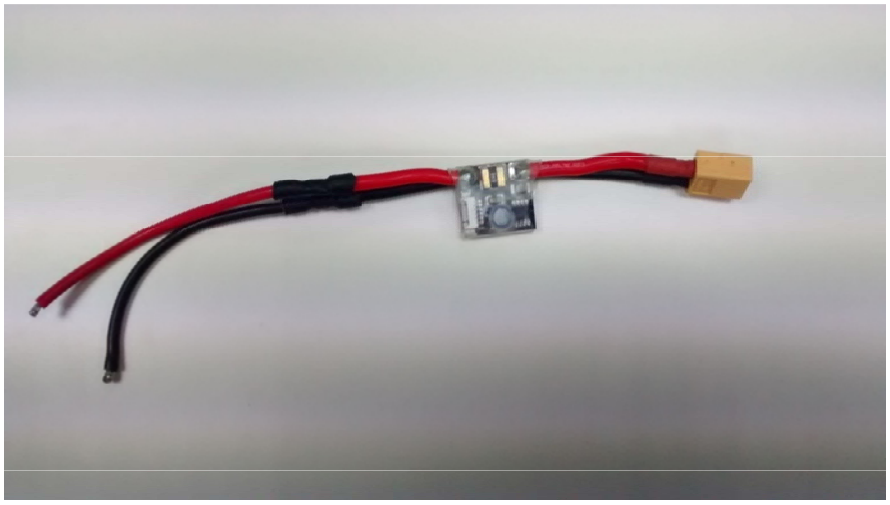


Fig 7: HK power module. Left side black and red cables are soldered to the frame back plate, which distributes power to the ESC's and the battery is connected to the XT-60 connector (Yellow). Six pin micro JST male connectors also gives current and voltage readings.

Table 5: Module specifications

Max input voltage	45V max
Max current sensing	90A
Switching regulator outputs	5.3V at 2.25A
Connectors	6-pos micro JST cable for data XT60 connectors to battery
Voltage and current measurement	0 to 5 V analog output. To be read by APMs 10 bit ADC.

1.2.4 Radio telemetry

To send data from our quadrotor to a ground station we need another radio transceiver. When purchasing them, country bandwidth availability should be checked first.

For most European countries 433 MHz version radios are appropriate. As most mobile communications happen over 900 MHz. [6]

These radios use MAVLink protocol to communicate [7]. This protocol is a very lightweight header only message system for small unmanned vehicles. Features include, dispositive identification and error correction up to 25%. It is also worth mentioning the handy python tools they have to create your own communication messages and that its open source.

Some of these specifications are listed on Table 6.

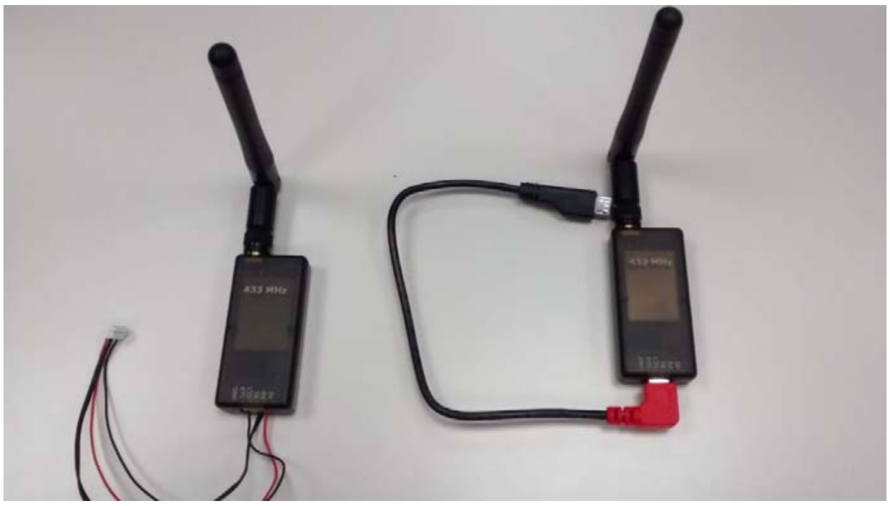


Fig 8: Radio transmitter and receiver (With antennas). Left, DF13 connector goes to the APM telemetry port. Right, connected via USB to the PC.

Table 6: Telemetry module specifications.

Supply voltage	3.7-6 VDC (from USB or DF13 connector)
Transmit current	100 mA at 20 dBm

Receive current	25 mA
Serial interface	3.3 V UART
Size	25.5x 53x11 mm
Weight	11.5g

Table 7: Telemetry module specifications

1.2.5 GPS

The Global Positioning System (GPS) will give the drone the ability to fly by itself. Autonomous flight is not something we intend from the start, but is something the drone is capable of doing.

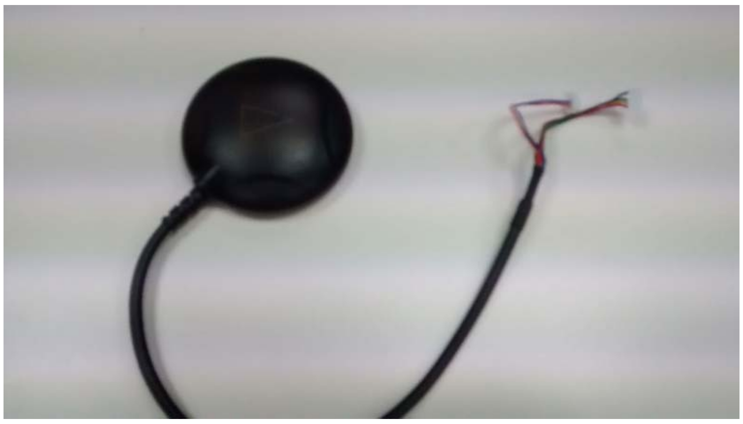


Fig 9: The GPS module, LEA-6H GPS, contains a GPS and a compass. The compass is very useful as we can separate the GPS module from noise sources like motors.

The GPS is connected to an exclusive port on the APM and the compass is connected to I2C buss as any other device.

Table 8: GPS module specifications and features

Data rate	5 Hz update rate
Antenna	25 x 25 x 2 mm ceramic patch antenna
Battery	Rechargeable 3V lithium backup battery
Memory	I2C EEPROM for configuration storage
Others	Power and fix indicator LEDs
Configuration	Baud rate 38400
Weight	38g with case
Size	27.5 x 27.5 x 7mm

1.2.6 Radio Controller

Even when the final objective would be to fly autonomously you always need a radio transmitter to control the drone. In Spain law referring to UAVs changes every month. In the meantime, only the pilots are responsible for a safe flight. Even if the drone is capable of autonomous flight we should always be ready to take direct control of it.

As an entry controller the best deal we could get was the Turnigy X9. It includes 2.4 GHz radio transmitter and receiver with 8 channels in PPM mode or 9 channels in PCM. PPM mode transmits data as PWM signals but in a much more packed signal. For a normal 20 ms PWM signal we would need 160 ms to transmit 8 channel data, in PPM 8 PWM channels are compressed on 20 ms. Therefore, the data rate is bigger and the response time lower. PCM on the other side is packing all the information on digital signals, for which you need a coder and decoder.

Table 9: Radio transmitter relevant information

Number of Channel	8ch ppm/9ch pcm
Buzzer	Yes
Low Voltage Show	Yes
Battery	3S
Display	128 x 64 LCD

1.2.7 Flight Controller

The most important piece of equipment you are going to need is a flight controller. As commented before we will need all the open source code so we can modify it and make it easy for us to use.

For this reason we went for the Ardupilot project [2] which is a controller board not only capable of flying and quad rotor but is capable of controlling all sorts of unmanned vehicle. Another big advantage is the massive community backing it up, a lot of different things have been done with this system around the world that will help us through our journey.

The source code and even the hardware is open sourced so we have different manufacturers distributing the Ardupilot boards (Fig 10). For the price, the

Hobbyking HKPilot Mega 2.7 (A slightly modified version of de Ardupilot APM2.6 board) is the best deal we could get.

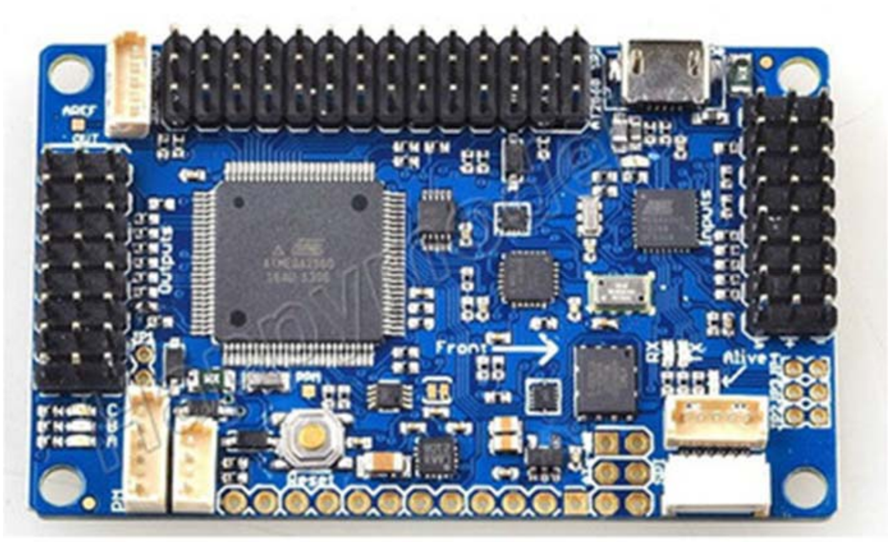


Fig 10: APM flight controller board. Manufactured by Hobbyking. The physical board colour is black

Table 10: APM table of specifications

Dimensions:	44x70x15mm with case
Weight:	32g with case
CPU:	Atmel ATMEGA2560 and ATMEGA32U-2
Logs:	4MB flash memory chip
Sensors:	• Invensense's 6 DoF Accelerometer/Gyro MPU-6000 • Accelerometer • High-performance Barometric pressure sensor MS5611-01BA03 • Honeywell HMC5883L-TR Digital compass
Interface:	• Micro-USB • External compass support • GPS input, I2C, Power module input • Telemetry radio, OSD and airspeed sensor ports • Analog/digital I/O pins.

Chapter 2: Software and Interfacing

2.1 Ardupilot and Arducopter

To start off, we recommend getting git to download and manage Ardupilot code. Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency [8]. Using git we download the code to a directory and keep track of changes or go back to previous versions of the code.

To build the code there are several platforms on each operative system. For each one the steps to follow are different. In our case, the easiest way to compile APM code is using APM's modified Arduino IDE and installing the AVR MHV tool chain (Fig 11). The process is explained on the APM developer's web page [9]. For extended use, we also recommend using Eclipse, linked to the Arduino compiler, but with all the advantages in terms of navigating through the code.

The basic structure of the APM is divided in 5 parts:

Vehicle directories: Here we can find every vehicle code, as we are going to use a quadrotor we would use the Copter code. They also include the make.inc file which lists library dependencies so you can build the code.

AP_HAL: The HAL (Hardware Abstraction Layer) makes possible to port Ardupilot to different platforms and hardware. With this layer making code for the APM supported boards it is much easier as the hardware components are handled by the HAL. For example, we will use the I2C communications later to connect our sensor, thanks to this layer we will have to configure the I2C bus semaphore and wait for the bus master (the APM board here) to request the data from our sensor.

Tools directories: All the miscellaneous support directories like tools for testing and log replay features go in here. We will not be analysing this section.

External support code: APM code also covers PX4 boards, Gimbal code for camera gimbals and mavlink for radio communication protocols. We will be moderately interested in the mavlink code here to send the sensor data to the PC or land station.

As the reader might or not know arduino is not multi-threaded natively, so when you need to have code running simultaneously you have to write your own scheduler code to manage it. APM based boards do this using timers and call-backs. One common use of threads is to provide drivers a way to schedule slow tasks without interrupting the main autopilot flight code [10].

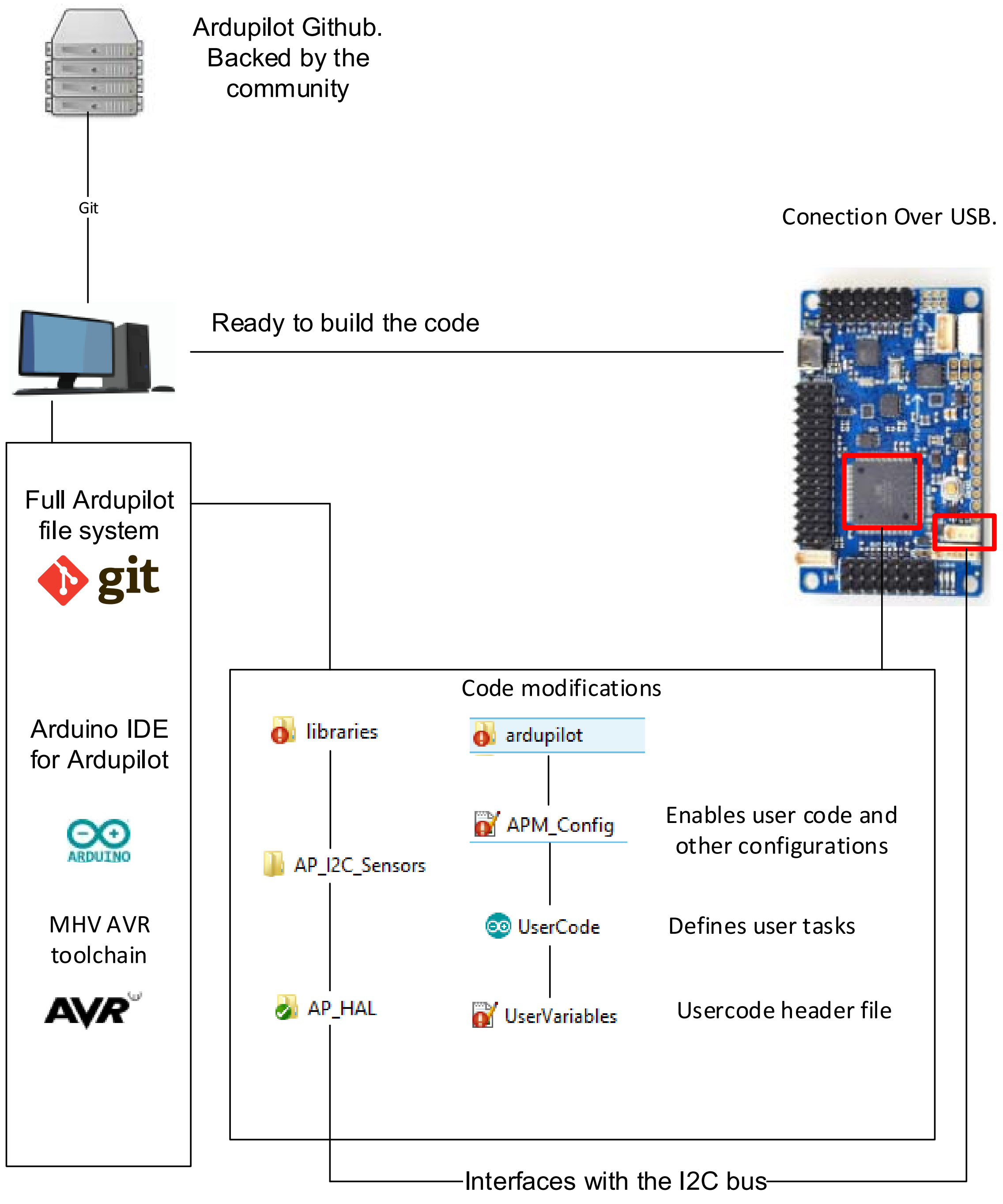


Fig 11: Explicative scheme of the hardware and software involved in the modification we did to the code. The Hardware Abstraction Layer (HAL) is the boundary library between code and hardware.

2.1.1 Writing a task

AP_scheduler controls the runtime on the code. It runs them on a priority system, attending those which are critical to the integrity of the vehicle first. After all stabilization or user input tasks are done (Fig 12), other secondary tasks, like logging data, are attended.

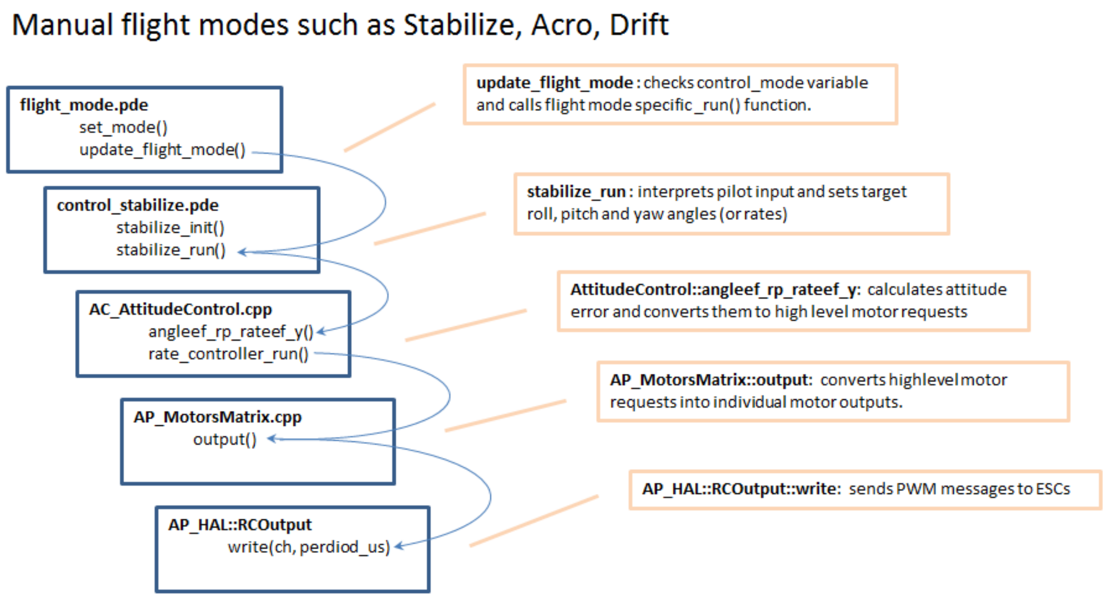

Manual flight modes such as Stabilize, Acro, Drift
flight_mode.pde
set_mode()
update_flight_mode()
control_stabilize.pde
stabilize_init()
stabilize_run()
AC_AttitudeControl.cpp
angleef_rp_rateef_y()
rate_controller_run()
AP_MotorsMatrix.cpp
output()
AP_HAL::RCOutput
write(ch, perdiod_us)
update_flight_mode : checks control_mode variable and calls flight mode specific _run() function.
stabilize_run : interprets pilot input and sets target roll, pitch and yaw angles (or rates)
AttitudeControl::angleef_rp_rateef_y: calculates attitude error and converts them to high level motor requests
AP_MotorsMatrix::output: converts highlevel motor requests into individual motor outputs.
AP_HAL::RCOutput::write: sends PWM messages to ESCs

Fig 12: Example code of the Ardupilot code workflow. It shows the tasks to accomplish to respond to user RC input. Starting by checking the flight mode [9]

The most important task a quadrotor controller must do is the attitude update. When new inertial measurement unit (IMU) data, which measures acceleration on all axis, is available the controller must quickly analyse it and reacts accordingly. This task will define the system ticks, the microcontroller is free to attend other tasks between ticks, but must update the copter attitude every 10 milliseconds.

Looking at the copter example we can see other important tasks are reading the RC radio input or checking and updating the throttle. The parameters required by the scheduler are, the name of the task, the frequency of calling (measured on system ticks) and the expected finish time in microseconds.

```
SCHED_TASK(rc_loop,            4,    130),

SCHED_TASK(throttle_loop,      8,     75),

SCHED_TASK(update_GPS,         8,    200),
```

We can add new tasks to run between ticks. The best way to do this with Arducopter is to use some pre-prepared task headers. These are called User Hooks.

To configure User Hooks we need to modify APM_Config.h file and uncomment some lines. We could also define our own headers there and create a task from blank but we would be doing just the same. Just as any arduino code our routine will consist of, at least, an initiation code and a loop code.

define USERHOOK_VARIABLES "UserVariables.h"

Needed to allow us to use the user hook headers file. The file UserVariables.h must contain the headers of our program just as any C program will contain.

define USERHOOK_INIT userhook_init();

define USERHOOK_MEDIUMLOOP userhook_MediumLoop();

Defines the user hook init and loop functions, doing this, the functions now exist and will be called by the scheduler.

The init function will be called once on start-up, should contain health checks to make sure the sensor is reading correctly before taking off.

And lastly the userhook_MediumLoop will hold the code running on loop. Here the Medium category is just a predefined frequency to call the task but Arducopter has a lot of different loops. For example, we are going to read radiation data every second, so we used the super slow loop for that.

define USERHOOK_SUPERSLOWLOOP userhook_SuperSlowLoop();

With this in mind we can write our task to register radiation data (Fig 13).

2.1.2 Logging data

Our APM 2.x has 4 megabytes of storage capacity on a Flash memory, accessible through an SPI interface. All the complexity of writing there is hidden behind an API, the API writes a log file for every flight and manages the wrapping when it fills up as well.

Data logging is expected periodically and should have a self-describing data structure. It is self-describing as the ground station will interpret the format and display it without the need of a common scheme to all the log messages.

```
#define LOG_TEST_MSG 1

              struct PACKED log_Test {

                LOG_PACKET_HEADER;

               uint16_t v1, v2, v3, v4;

               int32_t  l1, l2;

              };
```

The packets has a header for identification, afterwards, the data types are defined as we would declare them on any other C language.

```
static const struct LogStructure log_structure[] PROGMEM = {

 LOG_COMMON_STRUCTURES,

 { LOG_TEST_MSG, sizeof(log_Test),

 "TEST", "HHHHii",        "V1,V2,V3,V4,L1,L2" }

};
```

We first define a data structure containing 4 unsigned 16 bit integers and 2 signed 32 bit integers. This message is stored in program memory due to hardware constraints. The log would start with “TEST” and display the data separated as we did in the log structure. Data type is also present in the message, after the header, a letter for each data type is written between quotation marks. “H” is for 16 bit integers and “i” for 32 bit integers.

2.2 I2C connection with Arduino

When using Arduino some problems appeared, these will be discussed in the fourth chapter.

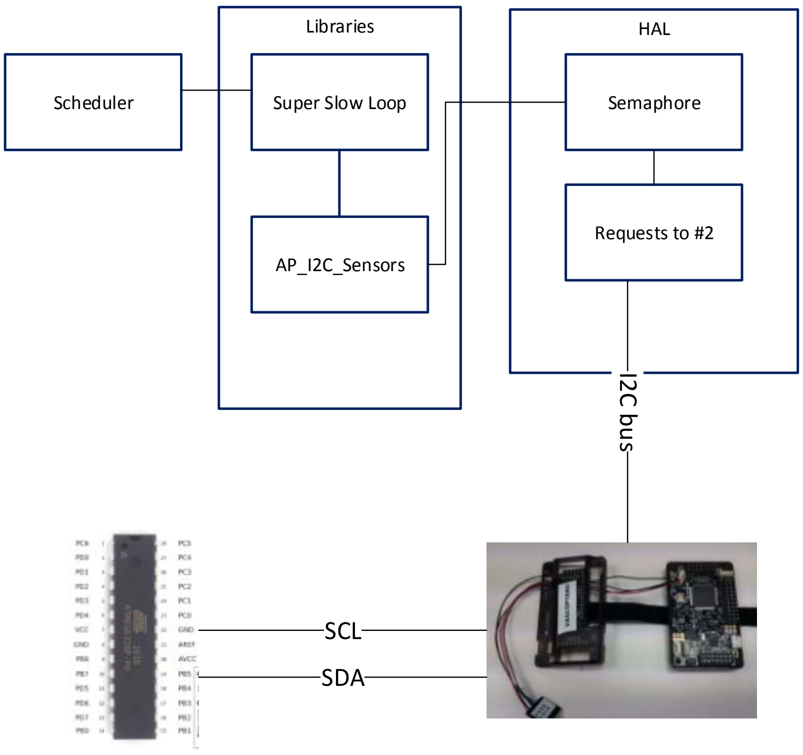


Fig 13: Basic scheme of the code. The Scheduler calls the super slow loop. On the loop code our "sensors" object has the routines needed to perform the readings. The task calls the HAL for a semaphore object to request the I2C bus and once the bus is clear for use we are connected to the Geiger arduino chip.

We decided to use an Arduino 328p chip to acquire the data and communicate with our APM. The code used to communicate an Arduino to APM and the code needed to read any other I2C sensor is very similar. So the work done here could be transferred to other applications.

APM I2C libraries are based on Arduino (Wire library) and so we should not have any issues. To transfer data we configure the Arduino as a slave and give it a name on the bus:

```
Wire.begin(2);
```

*Wire.onRequest(**requestEvent**);*

We named it #2 as we know it’s a free name on this board. Other I2C devices have their own name and should be checked accordingly. We then write a routine to attend when the slave is requested on that address:

*void **requestEvent()**{*

*Wire.write((uint8_t *)&data, 12); }*

On this case, the routine sends twelve data bytes to the APM that contain the measurements of our sensor. There are 12 data packets because we are sending 3 double numbers, each one is 32 bits long (4 bytes) and the packets are sent in bytes.

2.2.1 I2c connection to Ardupilot

AP_HAL Semaphores use whatever semaphore system is available on the specific platform, and provide a simple mechanism for mutual exclusion. For example, I2C drivers can make sure only one device is using the bus. [11]

On our APM board the I2C bus master is the APM. This cannot be changed, and the bus control should be managed by the flight controller to ensure it does not get blocked. As I2C protocol can handle addresses we can have several devices connected to the same bus. Also, I2C uses 2 wires for connection SDA and SCL signals. One transmits the data (SDA) and the other one is the clock source (SCL) (see Fig 13 and Fig 22).

Our APM uses 5V for any other I/O port excluding the I2C bus, which runs at 3.3V using logic level conversion units. This is because modern GPS modules run on 3.3V technology. Although, I2C can be used with different voltage devices, for our sensor we should aim for a 3.3V interface.

Chapter 3: Sensors

3.1 Radiation sensor

3.1.1 Geiger-Müller tube

For a Geiger-Müller (also known as just Geiger or G-M) tube to work it requires high electric fields to be applied to its terminals, these fields may vary from 400 V up to 1000V. Due to these high voltage electric fields when radiation collides with the gas inside the tube an electron is stripped from a nucleus, rapidly accelerating and colliding with other particles in its way. These other particles can then return to its ground state in no more than a few nanoseconds emitting photons in the process [12]. These photons might also ionize other not tightly bound electrons and maintain the chain reaction.

One mayor disadvantage of the G-M tubes, apart from the fact that we cannot distinguish between radiation types, is the relatively large dead time period after a discharge has happened (Fig 14), for example, a SBM-20 tube has around 200 microseconds of dead time, and especially for high count rates, this phenomenon should be taken into account. The process that terminates the tube discharge is led by the positive ions created in this cascade effect. The heavy ions cannot move as fast as the electrons, so, as the cascade effect develops in the tube positive ion concentration increases, leading to the diminution of the electric field present in the tube. Eventually no more cascades are created as the field drops below a certain threshold and the discharge is finished.

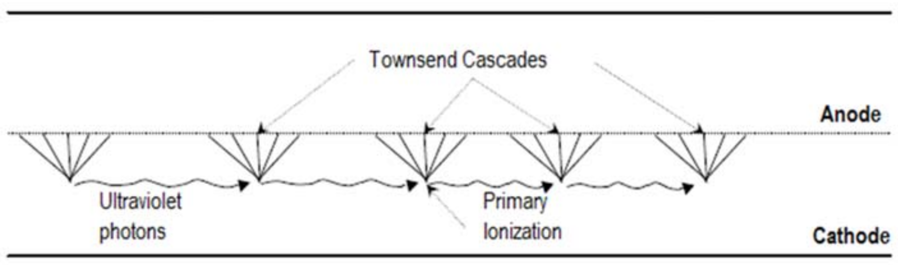


Fig 14: Cascade effect inside a Geiger tube, the fast photons trigger the discharge on the whole tube.

If only one gas is present in the tube there would be high chances of false second pulses after a real one occurred. When the initial discharge is terminated the heavy ions slowly drift towards the cathode (outer wall) to recombine with a surface electron. As the Ion and the Electron recombine to reform an atom, which is a more stable lower energy state, a quantity of energy equal to the ionization energy minus the energy needed to extract and electron from the cathode surface is liberated. The liberated energy could have enough energy to extract another electron from the cathode's surface (photoelectric effect), and thus, a second cascade effect would begin. A quenching gas is the second gas filling the G-M tubes, this gas composes 10% of the total gas filling but is enough to stop second cascade processes from happening. These gases can be large organic gases (which decrease useful tube life time) or halogens. For example our selected (Fig 15) tube contains Bromine as the quenching gas, so we will not need special treatment for the double cascade problem.

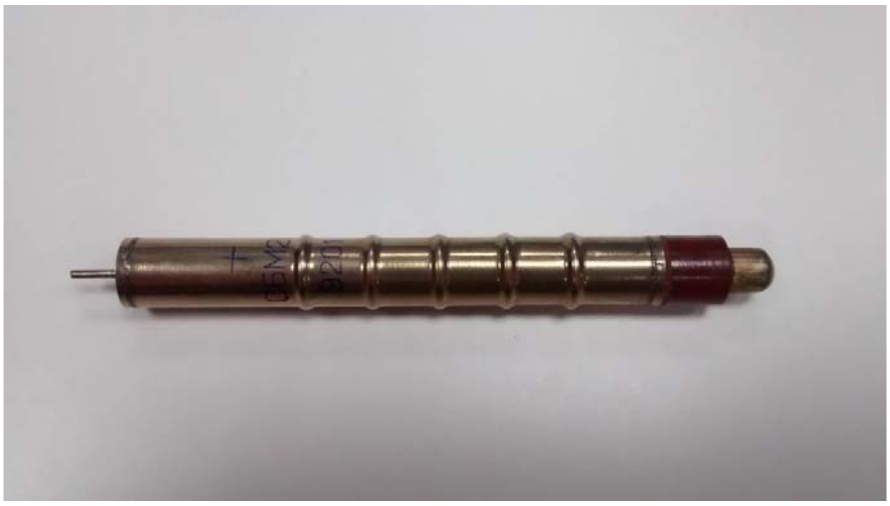


Fig 15: Image of the SBM20 a Russian Geiger-Muller tube. To the left the small tip of the Anode is isolated from the external wall (Cathode).

As for the output pulse of the tube, we expect a rapid rising edge, typically around microseconds, followed by a slower rise. The first rise corresponds to the first interaction cascades but chain reactions will trigger them all around the tube more slowly. The absolute maximum of the pulse would require a lot of time and does not give any special information so the sensing circuit time constant is kept low (less than 100 microseconds) to leave only the fast leading edge of the pulse. [12]

Lastly, the operating region of a G-M tube has some set-up advantages. If voltage is below the threshold it will not measure anything, but voltages bigger than the *starting voltage* will make it work. Just after surpassing the stating voltage a plateau region exists, in this region, typically flat, max counting rate is achieved for a constant emission test source. Elevating the voltage further may cause damage on the tube as constant discharges are triggered, but for most tubes the plateau region (Fig 16) will be quite wide which makes G-M tubes quite resilient to sub-optimal operating conditions.

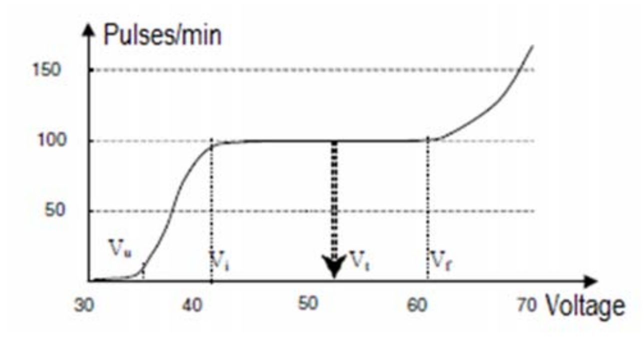


Fig 16: Geiger tube plateau region. Dashed arrow indicates the operating voltage for this example.

3.1.2 Boost voltage generator

To generate up to 1000 Volts we will need a boost converter type generator, where we will transform the 5 volts given by the power module.

There are plenty of boost converter or flyback topology based voltage generators, open source on online resources. We will not be doing anything new here, but the online resources do a really poor job of explaining why their circuit works.

At first we did not know what tube we would be using so we searched for similar designs and went for one that let us control the output voltage by simply adjusting a potentiometer (more on this on the conclusions).

Booster type generator design can be very complicated. The high voltage will be generated by an effect called inductive kickback. Conducting a current through and inductor will make it store energy in magnetic form (Fig 17).

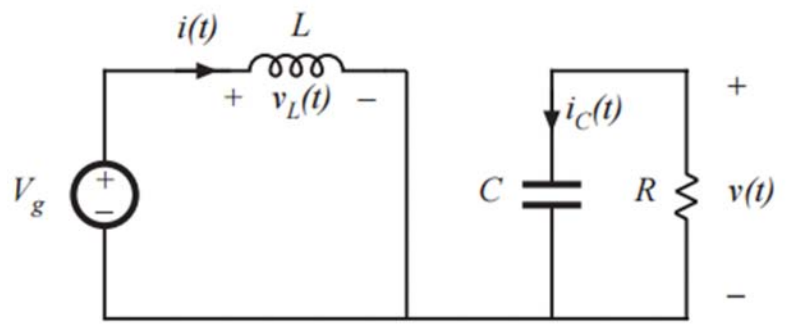


Fig 17: Booster circuit approximation on the “on” phase. Current increases on the inductor as it is connected to the battery.

Cutting the current in the inductor opposes this change and the magnetic energy stored in it is converted into current with a very high voltage peak. (Fig 18).

$$V_L(t) = Vg - v(t)$$

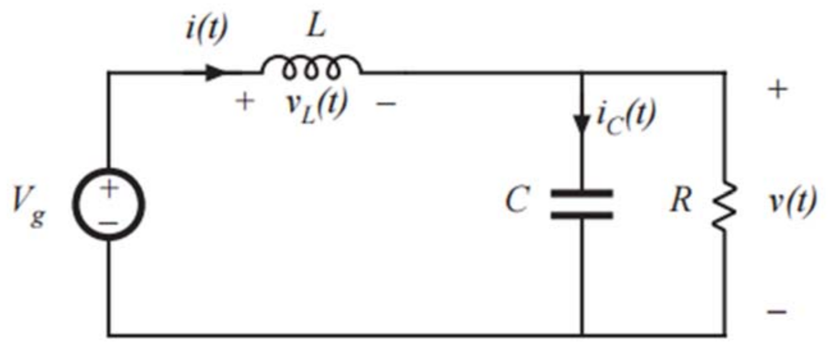


Fig 18: Booster circuit approximation on the “off” phase. After opening the switch to ground the inductor opposes the change and creates a big voltage spike to maintain current on the inductor. This spike charges the capacitor as V_L is negative on this phase.

We then store this voltage peak in a capacitor which will feed our load as we can see on the following image (Fig 19).

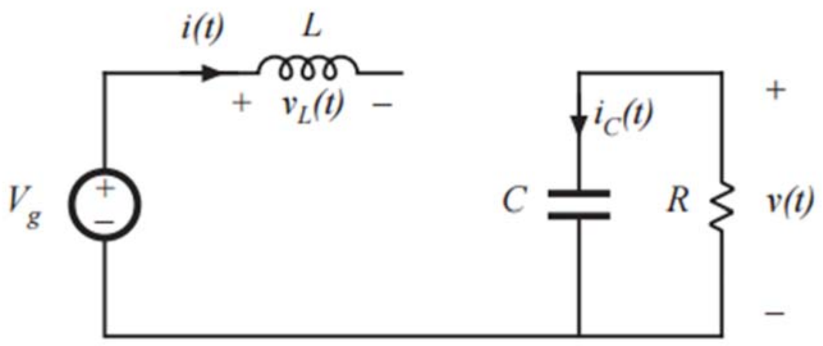


Fig 19: Booster circuit approximation on the "off" phase. The inductor current decreases over time until there is no more energy on it. Any load current is fed by the capacitor until next battery switching cycle begins. The open circuit is physically realized using a diode so current can only flow forwards.

For around 1000V peak with a 5V power source we will need a 15 mH inductor and 25 mA peak inductor current. This value is purely experimental. Simulations with PSPICE give us a good idea of how high the peak can be, but even on simulations, having a precise response of the inductor is difficult and so we will rely on previous works on this matter to select our components. [13] [14]

Boost converters to supply such high voltages will be used in Discontinuous Current Mode (DCM). What is less well known is that when the load current is low enough, a flyback converter behaves as a constant volt-age source. It converts the peak inductor current to output voltage in a ratio that depends only on the inductor and the capacitance associated with it [15]. So we will be using this peak inductor current controlled by a potentiometer to regulate the voltage.

For that, a transistor working as a switch will be used. This transistor is a power transistor capable of isolating the high voltage spikes that the inductor will generate. Transistor polarization voltage is also considered here, in this case, it has a 0.55V gate threshold (see part list).

$$V_{source} = 5V, V_{transistor} = 0.55V,\ V = IR$$

$$R_{trimmer} \rightarrow between\ 22\Omega\ and\ 100\Omega$$

$$I_{peak} \rightarrow between\ 25mA\ and\ 5.5\ mA$$

To make sure we can supply the Geiger counter we assume a 1000 counts/second maximum count rate and we calculate the power consumption of the Geiger tube (Calculations are done for the SBM20).

$$SBM20\ recomended\ voltage = 450V$$

$$SBM20\ Capacitance = 5pF$$

The energy stored by the tube then is,

$$E = \frac{1}{2}CV^2 \tag{1}$$

And its charge,

$$V = \frac{Q}{C}\ ,\ Q_{discharge} = V \cdot C = 25\ nC \tag{2}$$

Using the count rate from eq1,

$$I_{SBM20} = \frac{Q}{T} = 25\mu A \tag{3}$$

We then calculate the power consumption of the tube,

$$P = V \cdot I = 450\ \cdot I_{SBM20} = 1.25\ mW \tag{4}$$

Provided that efficiencies around 80% are not rare on these type circuits we will try to adjust the switching so that it gives 1.5 mW of input power.

As we already know that peak inductor current should lie between 5 mA and 25 mA, we choose the period of the switching:

$$T = 0.3\ ms\ , I = I_{peak} = 8mA^*$$

$$P_{Input} \approx \frac{1}{2}\frac{L \cdot I^2}{T} = 1.6\ mW \tag{5}$$

**It is around one third of the value and should therefore equate to around 300 V and should be near our worst case scenario.*

So we will be consuming less than 0.5 Amperes from the drone battery.

The load capacitor is also chosen to hold much more charge than the requirements of several Geiger tube counts. This capacitor is also the separation between 1000V and ground, so specially designed capacitors are required (see part list). Worst scenario for the load is to detect a particle every 200 microseconds (Geiger tube dead time). Using that and the charge per event calculated on (2).

$$\frac{T}{T_{dead}} \cdot Q_{discharge} \approx 0.4\mu C$$

Around tenfold bigger capacitor for 450 Volt reference load and commercial capacitor values sets us on 10nF capacitors.

Lastly, in order to reduce the voltage ripple associated to booster circuits, there is an RC filter with cutting frequency of 15 Hz to maintain DC power as close as we can.

$$f_o = \frac{1}{2\pi RC}, R = 1M\Omega\,,\ C = 10nF \tag{7}$$

To layout the circuit and print the PCBs we used KiCad open-source design tools. [16]

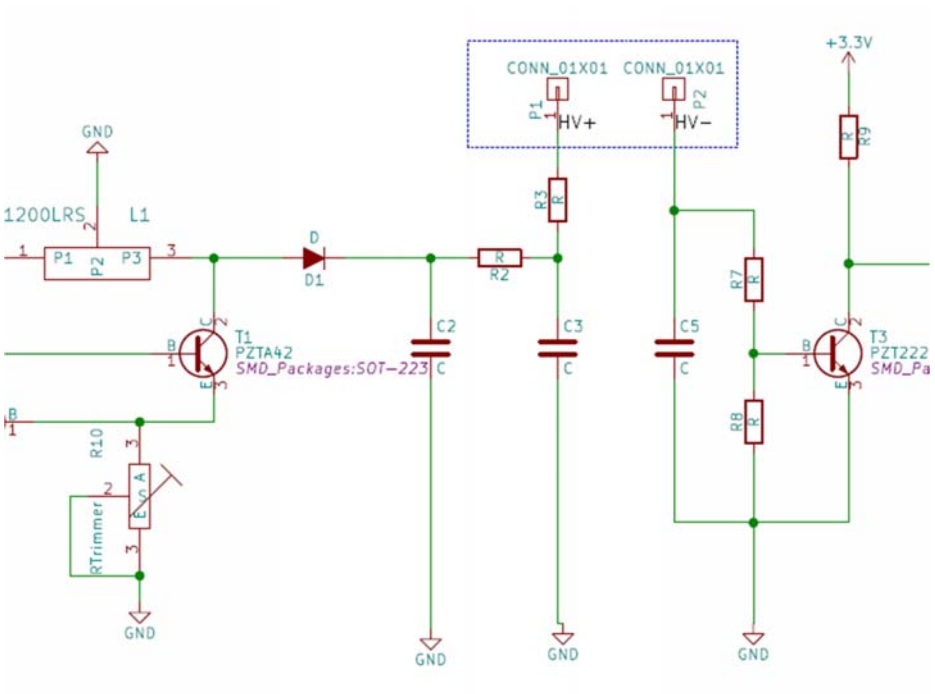


Fig 20: Booster section of the circuit. T1 transistor is switching every 0.3ms to deliver power onto the load capacitors, C2 and C3, which also conforms an RC low-pass filter with R2 to reject the voltage ripple. The tube is connected to P1 and P2.

To both sides of the tube P1 and P2 here we also need some resistors. For optimal tube operation R3 (anode resistor) and the equivalent of R7/R8 (cathode resistor) should have a 45:1 Ratio, at least that seems to have become "industry standard". [13]

The cathode side (P2) is just a voltage divider. When a particle is detected a 100V spike is created. The capacitor protects the transistor and the voltage divider steps it down to around 1,5V. The time constant of this circuit should be kept below 100 microseconds to avoid pulse stacking.

$$V_{transistor} = 100V\frac{R8}{R7+R8} = 1{,}478V, R8 = 1{,}5k\Omega\,, R7 = 100k\Omega \tag{8}$$

Note that the diode not only needs to withstand high voltage, it should also be a fast recovery diode so that current does not reverse.

Table 11: Table of components

Name	Value	Mouser No.
Inductor (L1)	15 mH	580-12LRS156C
Trimmer (R10)	0-100 Ω	652-3299W-1-101LF
FR diode (D1)	1 kV and 75 ns	625-BYG23M-E3/TR3
Capacitors (C2,C3)	10 nF and 1kV	77-VJ1206Y103KXGAT5Z
Capacitors (C5)	330 pF	Had spare
Resistors (R7,R8)	100 kΩ,1.5 kΩ	667-ERJ-P06F1003V
Resistors (R2,R3)	1 $M\Omega$,4.7$M\Omega$	660-HV732BTTD1004D
HV Transistor (T1)	1k2 V Bipolar NPN	511-STN0214
Transistor (T3)	0.55 V Gate	512-MMBT4401

3.1.3 Switching timer

With the previous calculations we know the switching frequency should be around 3.4 kHz or have a period of 0.3 ms. In this case it was realized with a 555 CMOS (lower current draw) Timer. Other options will be discussed on the build section, but at the moment it was the easiest solution.

The time the inductor needs to stay in tis on state is unknown (we can adjust it with the potentiometer). We let it "self-regulate" the on time by connecting the T2 voltage sensing transistor to the Reset (pin4 Fig 21). When T2 triggers current flows through

R4 and drives Reset LOW. This forces the output (pin 300) to go low fast through the discharge pin, which is grounded. [17]

When the output goes low T1 stops the current through the potentiometer R10 and thus C4 starts to discharge through R6 slowly. This RC circuit should retrigger the circuit after 0.3 ms, as was calculated in the previous section (eq 5).

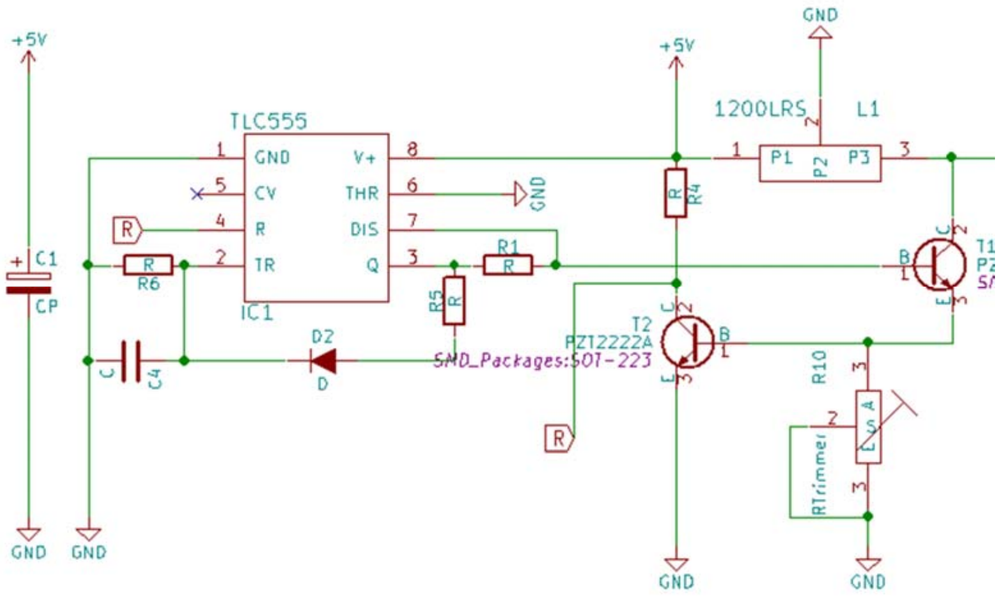


Fig 21: The TCL555 controls the switching of the T1 transistor. T2 triggers when the desired peak current is achieved, resetting the 555. C4 discharges through R6 until the 555 gets triggered again, causing Q to go high again and recharging C4 through D2 very fast.

So the RC oscillator,

$$V = V_0 e^{\frac{-t}{RC}} => T_{trigger} = 213\ us \qquad (9)$$

The voltage comparator on the 555 triggers at 1/3 of the supply voltage. The capacitor is loaded through D2 diode so the starting voltage is 4.4V. (Fig 21)

Table 12: Timer components

Name	Value	Mouser No.
Filter Capacitor (C1)	220 uF	5985-AFK10V220-F
Diode (D2)	General Purpose	863-MMSD4148T3G
Capacitors (C4)	1 nF	77-VJ0805Y102JXJPBC
Resistors (R6)	220 kΩ	667-ERJ-6ENF2203V
Resistors (R5)	330Ω	667-ERJ-6ENF3300V
Resistors (R1)	1k Ω	667-ERJ-6ENF1001V
Sensing Transistor (T2)	0.55 V Gate	512-MMBT4401
Resistors (R4)	100 kΩ	667-ERJ-P06F1003V

3.1.4 Microcontroller

We used the Atmega-328P for this project because it was the first one available. To detect the tube events we just need a digital I/O port and an interruption routine. Annexed to this document should be the full program written for arduino to detect a falling edge.

The microcontroller is running off of its own RC oscillator. The configurations needed to achieve this are beyond the scope of this project. We will only discuss why this not a problem and the benefits on the build chapter. To do it we need to burn a new boot loader on the microcontroller, this is achieved with the Arduino IDE and some help from their forums and AVR fuse calculators. [18] [19]

The connections to APM and the two leds used to create patterns when initializing and after a detection are on the image below (Fig 22).

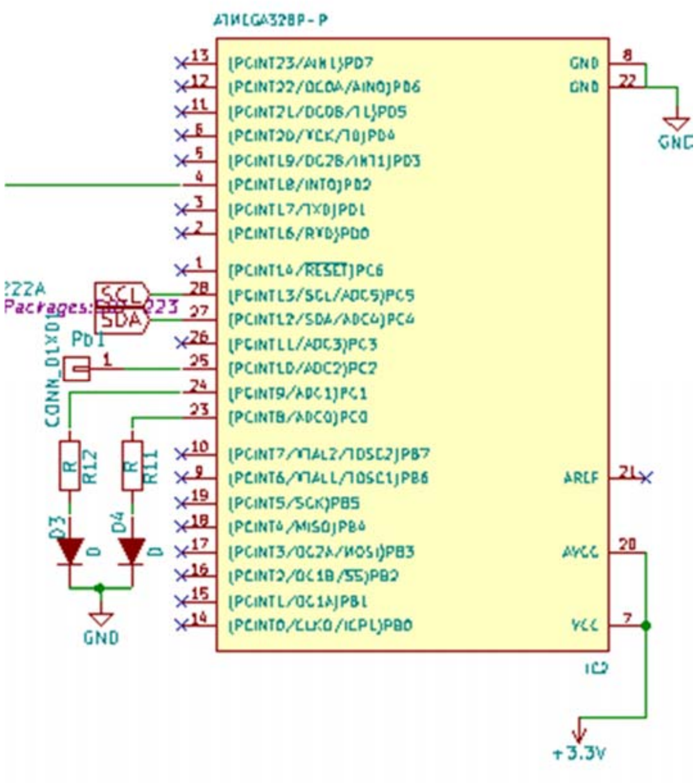


Fig 22: Atmega328P circuit connexions. 24 and 23 connect to leds so we can monitor the detections without a PC. 28 and 27 are connected to APM via I2C protocol. Finally pin 4 is used to detect the falling edge.

Using an external microcontroller is advised. Even when the task itself is very simple and our flight controller could easily handle it, it requires an external interrupt on the highest priority to make sure we do not lose any events in our measurements.

3.2 Measurements and calibration

The circuit was tested in the lab before going to calibration. We can clearly see the voltage is being generated by the inductive kickback on the oscilloscope by soldering a 1GOhm resistance in series with the oscilloscope. The oscilloscope probe was set to 10 MOhm resistance so the readings where near 100 times smaller.

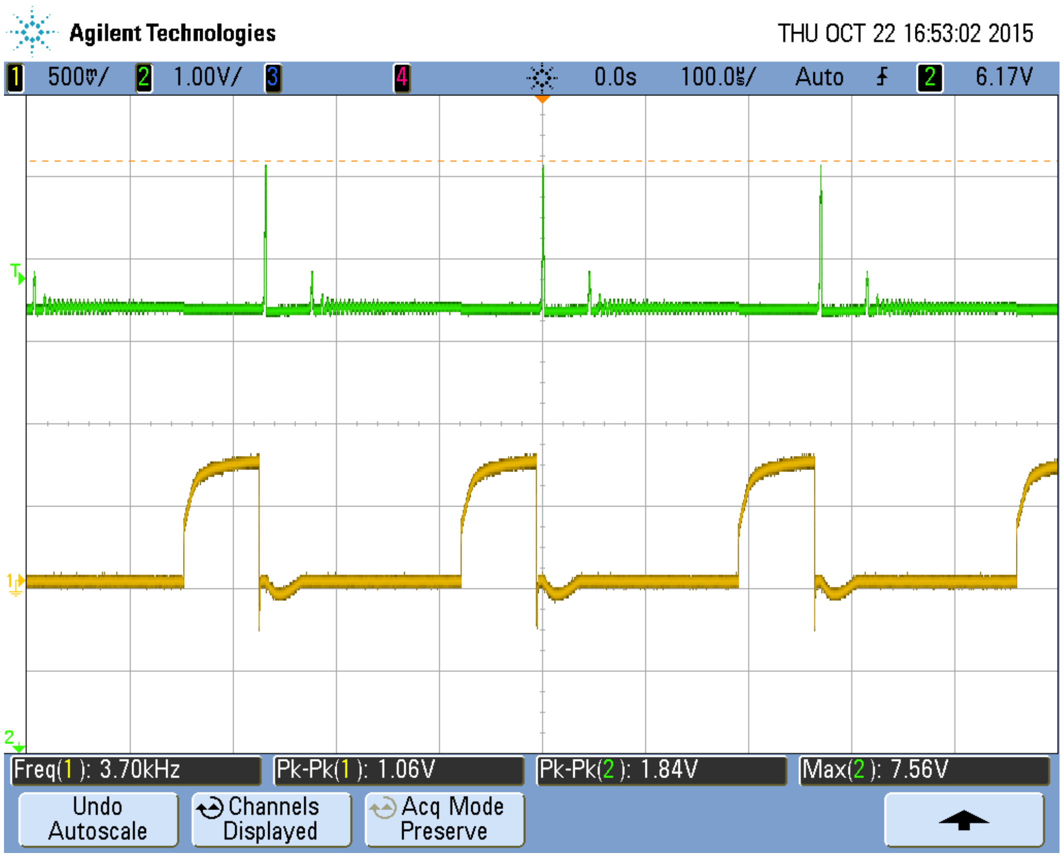


Fig 23: Measurements done on the Geiger circuit. One probe (yellow) measures the voltage on the switching transistor gate (T2). The green signal is measured through a 1GOhm resistor to dampen the signal 100 times. Once the transistor shuts inductor supply we see an immediate voltage kickback.

Setting the oscilloscope to single trigger and adjusting it just below the supply voltage lets us take a photo of the moment the Geiger tube registers an event. The inductive kickback occurs on every transistor switching and peaks at 756 V. After each kickback the voltage drops to its nominal level around 600 V in rms. The peak represents less than 1% of the total signal which is easily endured by the Geiger tube.

When an event is detected the voltage will drop significantly. The last resistor before the tube (R3 on Fig 20) and the RC circuit after the tube (C5, R7,R8) coupled with how much energy we are able to transfer per switching cycle define the time of recuperation. As it was stated before, these values are experimental and the “industry standard” is to use 45:1 anode to cathode resistance ratio.

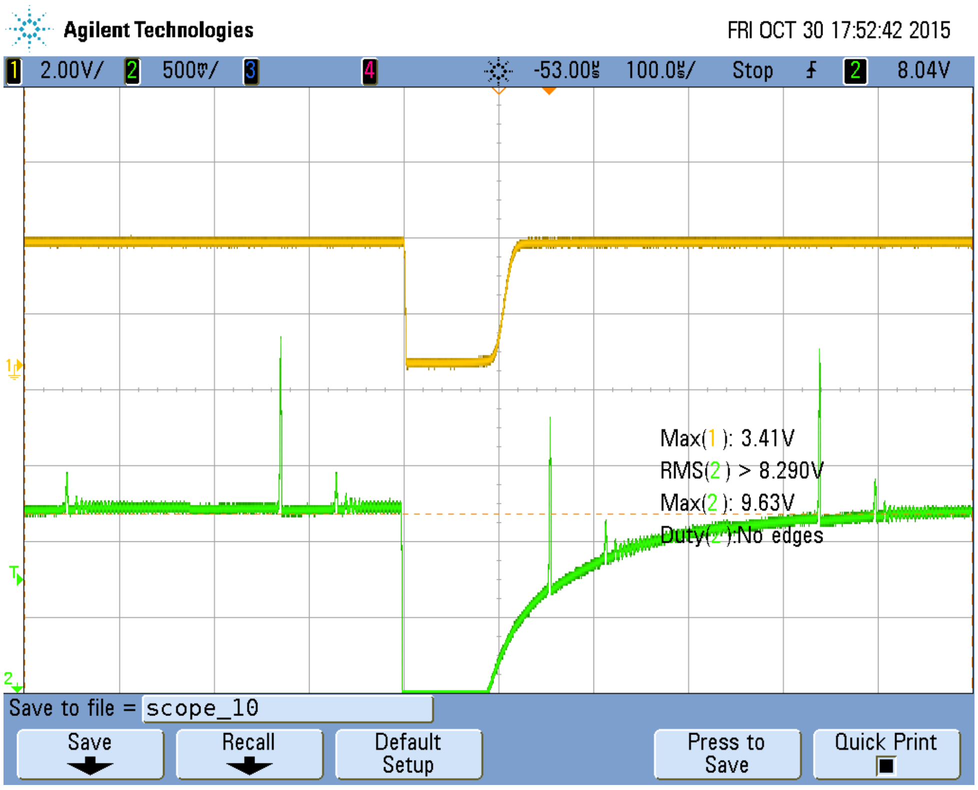


Fig 24: Radiation event capture. The green signal probe is connected to a 1GOhm resistor connected to the tube. The yellow probe is measuring the voltage drop on the other end (T3 collector). The event detection gives us a clean falling edge that will be detected by the Arduino. High RMS voltages where needed for the testing tube, we suspect malfunction.

The recuperation time, estimated around 200 us (see Fig 24), matches the tube dead time (detailed in section 3.1).

The SBM20 is an old Russian tube manufactured in mass during the Cold War. Once the voltage generator is working we need to characterize our tube.

To do this we went to the Radiation and Calibration laboratory at Instituto de Fusión Nuclear in Universidad Politécnica de Madrid. There and with the kind help of the laboratory technicians we tested the circuit.

3.2.1 The Plateau

As explained in section 3.1.1 we need find the plateau of our tube to use it as our working conditions. To do this, we vary the voltage of the circuit and measure count rate. The samples we used is called Nuclear B and Radium 226 and we had two tubes to test.

Table 13: Plateau calibration measurements

Tube 1(Nuclear B)		Tube 2 (Radium 226)	
Voltage (V)	Counts (CPS)	Voltage (V)	Counts (CPS)
745	11.6	615	298
755	11.97	680	323
835	12.08	730	334
915	12.49	780	333
1015	13.5	830	334
1115	14.8	875	338
1215	14.3		

On the table above we see the most remarkable data from the tests. We concluded that Tube 1 was not working well, as we were unable to find a plateau. Tube 2, on the other hand, behaved as expected. We suspect that Tube 1 might have been damaged because all previous work on the voltage generator was done and tested with it.

With this data we determine plateau and can choose the best working voltage as 780 V (the middle point).

3.2.2 Calibration

To fully characterize a Geiger-Müller tube we would need calibrated samples and a geometrical analysis to calculate the solid angle of incident radiation on the detector, methods which are complicated and are unnecessary for our detector and its purpose.

Radiation events are very low probability events. This type of events follow a Poisson's distribution and the error associated to it.

$$p(n) = \frac{(N)^n e^{-n}}{n!} \qquad (10)$$

Where N is the number of expected counts (the mean value we are going to measure) and so the standard deviation or error is $\sigma = \sqrt{N}$. When n tends to infinity it is just a normal distribution. This means we either need to measure for a long time or have high number of events to make a precise measurement Table 14.

Table 14: Relative error table

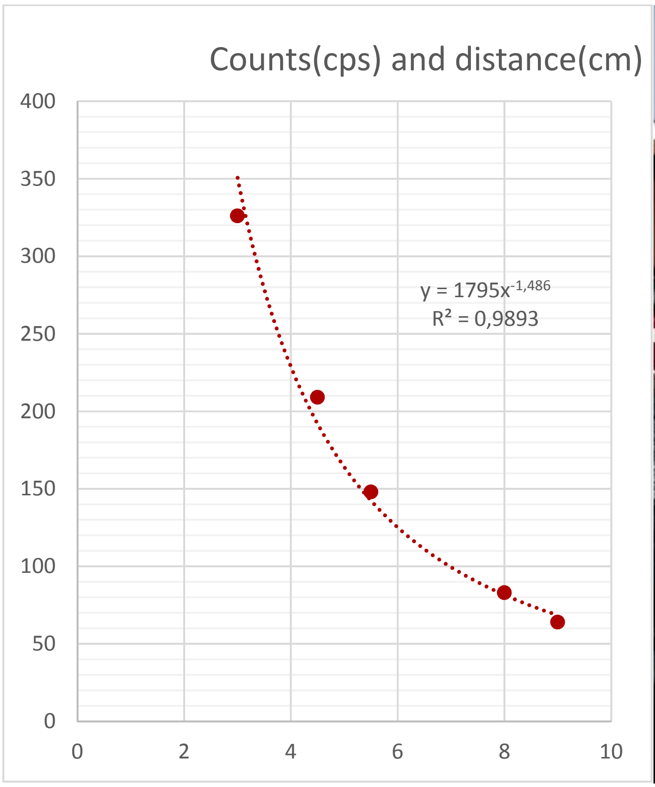
100 events in 10 minutes	$\frac{100 + \sqrt{100}}{10} = 10 + 1 \rightarrow 1\%\ error$
10 000 events in 1000 minutes	$\frac{10000 + \sqrt{10000}}{1000} = 10 + 0.1 \rightarrow 0.1\%\ error$

Knowing the reduced flight times we would aim at quick measurements of zones of high intensity radiation, for example, after a big disasters like Fukushima to quickly access a certain hard to reach area.

To make sure we are reading radiation simple test is to measure the count rate variation with distance. The following data (Table 15) was acquired with Tube 2 and Nuclear B sample:

Table 15: Radiation and distance measurements

Distance(cm)	Counts(cps)
9	326
8	209
5.5	148
4.5	83
3	64



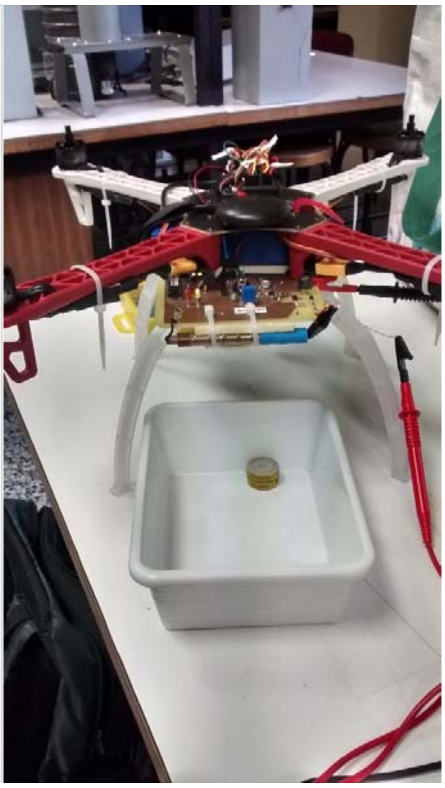


Fig 25: Left, graphic of the measurements taken at the UPM laboratory. Right, the drone (red and white) with the Geiger sensor board attached stays around 10 cm from the ground.

If we assume a punctual detector and source, the intensity of radiation should decay close to $\propto {}^{1}/_{r^2}$, where r is the distance to the detector (Fig 25). As we can see the curve is close, but there are variations due to solid angle between the sample and the tube, which is also not punctual, and the detector.

Chapter 4: Build steps

This section will be a little different. We will be reviewing the process followed to build and setup the quadrotor but we will also talk about problems encountered on the process.

4.1 Connection scheme

To connect everything up we followed the manufacturer guides [20] , as well as the board developer guides [21]. All the information is online on those resources and other various guides. Here we present our own scheme of connection (Fig 26).

We do not intend to make a step by step guide, but we summarize them here:

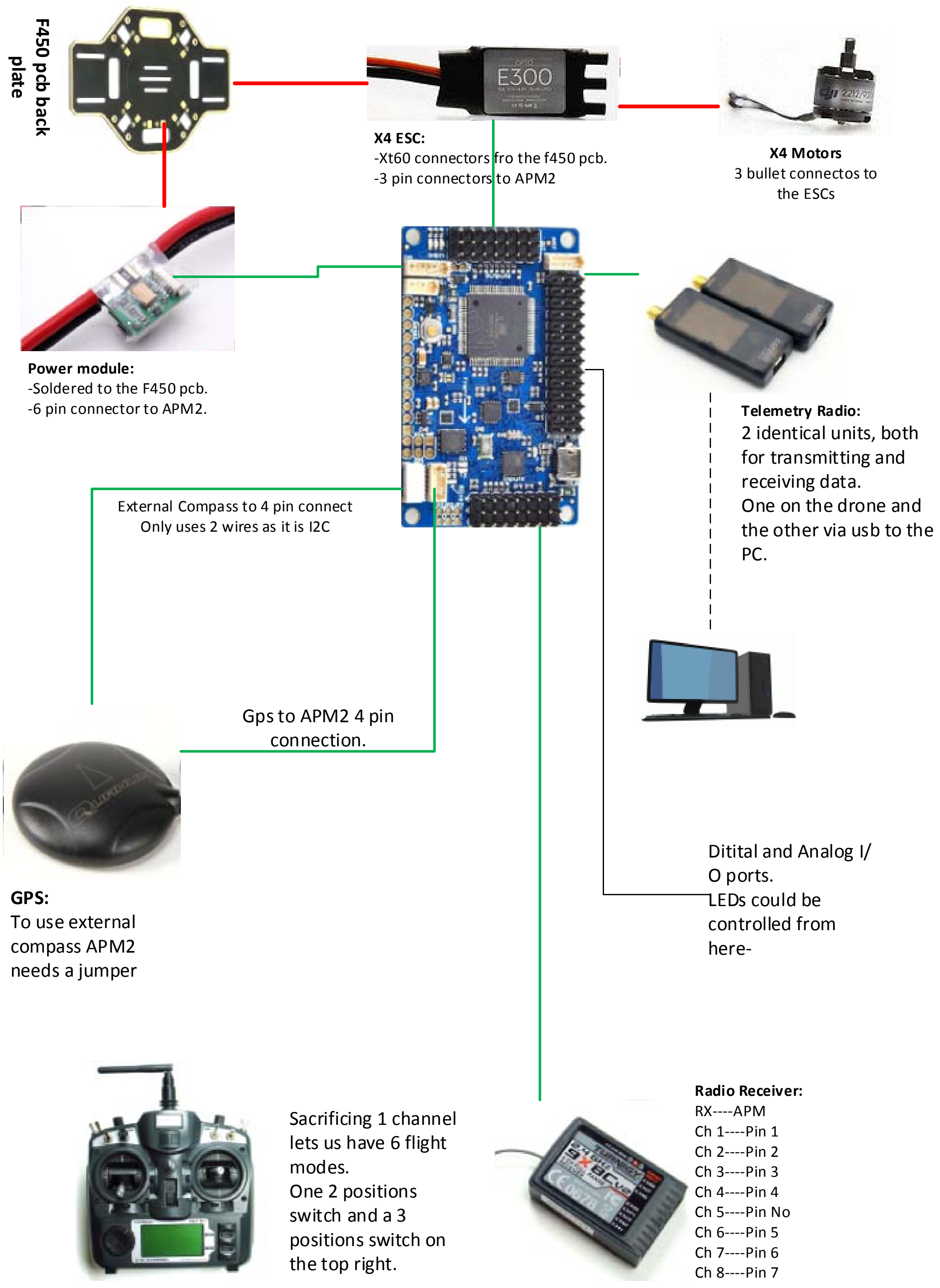
1. The build started by soldering the four power cables, like the ones the power module (Fig 7) uses, with XT60 connectors to the F450 back plate's four corners(Fig 26). The power module is also soldered here to the pads marked on the PCB. This conforms the power distribution circuitry.

2. After that, the F450 frame, assembled following their instructions.

3. The ESCs are tied with bridles to the frame arms, and the motors screwed in the end. Do not mount the propellers on the motors unless you are going to fly.

4. Radio receiver is mounted on the back of the drone with Velcro straps and the telemetry radio on the opposite side.

5. Finally the controller is tied with velcro to the top of the frame were we also place the GPS.



F450 pcb back plate

X4 ESC:
-Xt60 connectors fro the f450 pcb.
-3 pin connectors to APM2

X4 Motors
3 bullet connectos to the ESCs

Power module:
-Soldered to the F450 pcb.
-6 pin connector to APM2.

Telemetry Radio:
2 identical units, both for transmitting and receiving data.
One on the drone and the other via usb to the PC.

External Compass to 4 pin connect
Only uses 2 wires as it is I2C

Gps to APM2 4 pin connection.

GPS:
To use external compass APM2 needs a jumper

Ditital and Analog I/O ports.
LEDs could be controlled from here-

Sacrificing 1 channel lets us have 6 flight modes.
One 2 positions switch and a 3 positions switch on the top right.

Radio Receiver:

RX	APM
Ch 1	Pin 1
Ch 2	Pin 2
Ch 3	Pin 3
Ch 4	Pin 4
Ch 5	Pin No
Ch 6	Pin 5
Ch 7	Pin 6
Ch 8	Pin 7

Fig 26: Connections scheme for our drone.

4.2 Mission Planner

Mission Planner is a full-featured ground station application for the ArduPilot open source autopilot project. [22] As for our purposes, Mission planner lets us configure the drone, read Geiger radiation measurements and download the data logs of the flight and view them. Then main features we used for this project are described below. (Fig 27)

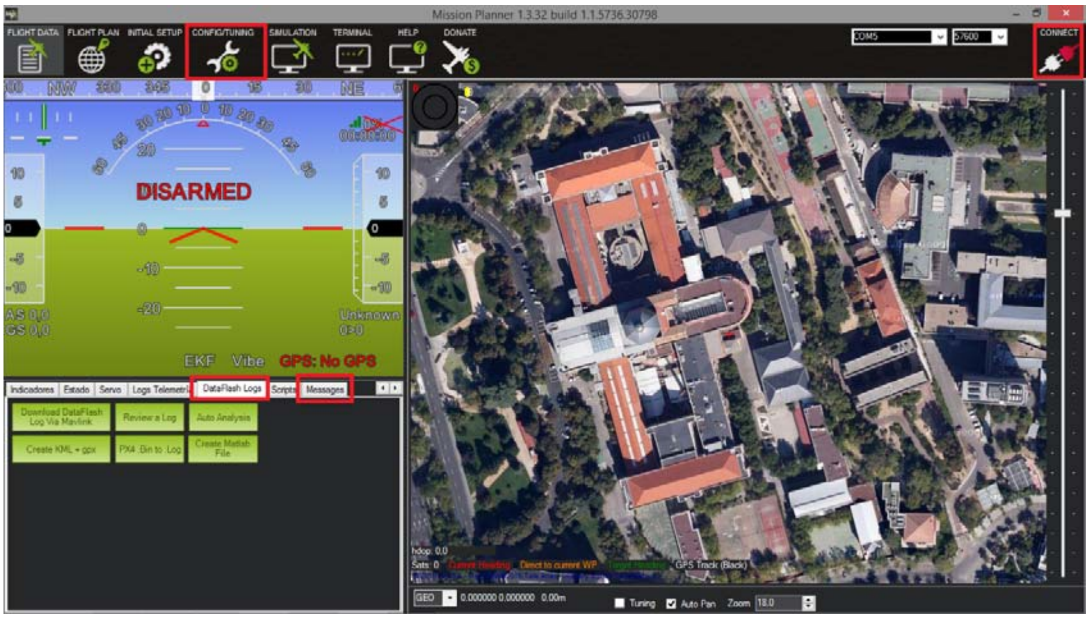


Fig 27: Mission Planner main window. The red rectangles highlight the features we are using.

Mission Planner can be connected to APM in two ways. The first one is using an USB micro cable, we used this mode to upload and test the code, and the second is using the telemetry radio.

Once connected through the telemetry radios and the drone powered up we will immediately begin to receive radiation data. We can check this data in real time through the Messages tab (Fig 27). Data is sent each second and shows the following format:

Table 16: Information display format

Cps:XXX	Total:XXX	Tmilis:XXX
Events on the last second	Total number of events	Time since operation on milliseconds

The DataFlash Logs tab lets us download the log files. These files contain all the sensor information as well as other vital data, like GPS data. With these logs we can even correlate radiation measurements with the position they were done from (Fig 28). Our sensor data is configured to appear under the “GEIG” header.

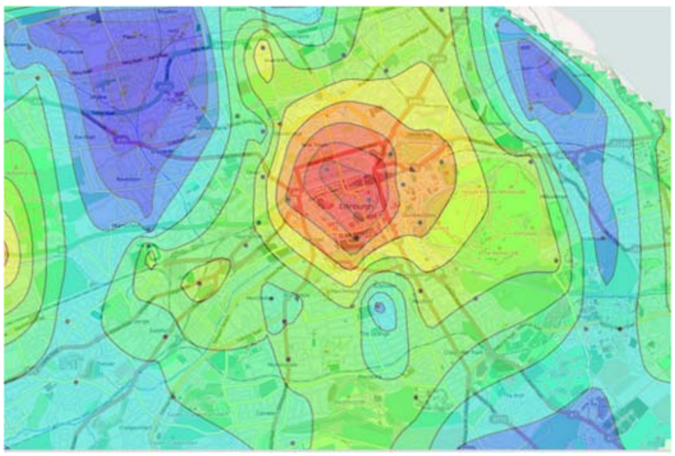


Fig 28: Illustrative heat map, radiation data could be gathered and mapped on heat map style presentation. The image is illustrative and does not contain real data.

4.3 Build Problem discussion

- To assembly the drone we used Velcro straps and bridles, this are very handy light weight ties. In the process we found ourselves re-doing certain parts of the drone and this method turned to be very convenient.

- Nuts and bolts should be tightened with epoxy glue to prevent them from untying on flight due to vibrations. This step should come near the end after the drone has been tested.

- The GPS mounting accessory is very fragile and it broke during one of the first flights. As we are using the compass located on the GPS capsule, locating it near the motors will make compass readings noisier.

- The Arduino IDE, provided by Arducopter and used for compiling the code works most of the time, but should be handled with care if multiple Arduino versions are being used. Some of our compilations where corrupted by using multiple instances at once on the same machine while testing code for the Geiger circuit (Standard Arduino) and APM Arduino IDE. We solved it by doing a fresh install.

- The Geiger sensor board did not work at first, with switching frequency changing at its will, we finally resolved it was caused by a cold welding on the circuit.

- At first having no crystal oscillator on the Geiger circuit seemed fine. We could use the 8MHz RC oscillator inside the arduino chip. This made sense because the I2C protocol has its own clock signal and it does not get affected by clock accuracy. Also, having the arduino run at lower clock speeds and 3.3V makes it consume less power and we can have some battery savings. The arduino microcontrollers that come with a P letter at the end of their name come with their own boot loader. This boot loader did not switch to the RC oscillator if an external clock signal was missing. To fix this we had to burn a new boot loader on the chip and change the fuse configuration to allow internal oscillator operation. [19]

- Our APM2 flight controller was a bit different from the official version as it was purchased from a different manufacturer. The I2C pins we are using to communicate with the Geiger sensor were not soldered to the physical port at first. This had very little documentation and was found on the Eagle files and soldered accordingly.

4.4 Complete parts list

Table 17: Parts list and prices for the project

Part	Description	Cost
Drone battery	Turnigy 5000mAh 3S 30C Lipo Pack (EU Warehouse)	35.72 €
Radio transmitter and receiver	Turnigy 9X 9Ch Transmitter w/ Module & 8ch Receiver	69.96€
F450 frame	DJI F450 +Propellers ARF kit	206.5 €
Geiger circuit parts and PCB (estimated)	Electronic components and PCB board	15 €
3 pin connectors for APM	Twisted 15cm Male to Male Servo Lead (JR) 22AWG (10pcs/set)	US$2.66
Additional propellers	Gemfan DJI Style Propeller 9.4x4.3 Locknuts (CW/CCW) (2pcs)	US$4.32
Propeller balancing kit	HobbyKing™ Universal Propeller Balancer, For T Style and Std Propellers	US$6.75
Other propellers	Turnigy Slowfly Propeller 10x3.8 Black (CCW) (4pcs)	US$5.60
Other propellers	Turnigy Slowfly Propeller 10x3.8 Black (CW) (4pcs)	US$5.40
Vibration damper foam	Anti-Vibration Foam (White Latex Foam)	US$1.47
Velcro straps	Battery Strap (5pcs/bag)	US$1.95
Battery charger	Turnigy E3 Compact 2S/3S Lipo Charger 100-240v (US Plug)	US$12.35
Battery charging safe bag	Lithium Polymer Charge Pack 18x22cm Sack	US$1.99
Handy battery monitor	HXT Simple Lipoly Monitor 2S~3S. A must have for all lipo users!	US$2.19
HK dampening mount	HobbyKing Universal Vibration Damping Mount (62x35mm)	US$3.99
XT60 connectors	Male T-Connector <-> Female XT-60 (1pc/bag)	US$1.25
Optional landing gear	DJI F450 /F550 Flamewheel Extended Landing Gear (4pcs/set)	US$5.99
Flight controller, GPS, Power module and telemetry combo	HKPilot Mega 2.7 Master Set With OSD, LEA-6H GPS, Power module, Telemetry Radio (433Mhz) (XT-60)	US$179.99
USD dollar to Euro conversion sum		214.71 €
Total		541.89 €

Chapter 5: Conclusions

In this work, we have assembled a drone as a platform for sensor applications and we have superficially characterized the main electronic devices and software considerations that a new user might need to know before venturing on the adventure of designing and building a drone. Firstly, we analysed an open source flight controller code. We have been able to isolate key software modules and modify them to implement our own application. The full flight controller code is very extensive and we quickly realized that trying to break down all of it was very complex and beyond the scope of this project. Secondly, we have studied how sensors interact with the systems available on drones and developed a Geiger-Müller sensor board for it. Drones are an ideal platform for sensor applications because they already contain a wide variety of sensors themselves.

By analysing the Arducopter code we were able to transmit custom data packets over radio to a Ground Station (a PC). These are recorded on log files during flight but can also be checked on line with the telemetry radio. This combination proved to be very useful, as we could do live test with real radiation samples. Even if radiation measurements over one or two meters are already a very difficult task, the quick development on precise autonomous flight could prove to be interesting in these applications. On this matter we have developed a testing ground for future work, the drone software and hardware are known to be capable of autonomous flight, but consistency needed to perform these tasks will only come through fine tuning.

Based on previous work we were able to break down and developed a lightweight Geiger sensor board for our drone. We tested various features, for example, the addition of a fuse for blackout prevention, with mixed results. Designing this Geiger sensor also showed us how extensive switching power supply field is, meaning there is a lot of room for improvement on this topic.

Last but not least, it is noteworthy that using commercial parts saved us money. Buying commercial parts is cheaper than building them one by one. This, combined with an open-source flight controller the final product is robust and reliable. The work accomplished here is also interesting for further research on sensor applications for drones.

5.1 Future perspectives

There are still some aspects that could require polishing or upgrading. Autonomous take off feature could easily be implemented with more time and testing on rangefinders. This matter would be especially interesting for mapping applications.

The possibility of mapping high radiation areas could be useful if the flight times where better, for example, by using better quality batteries or doing whole new builds with more efficient rotors and propellers. Radiation sensitivity could be improved by using an array of Geiger-Müller tubes, meaning the drone would have to stay less time in an area to do a more precise measurement.

The Geiger sensor board we used is a prototype. The size can be made much smaller and some components are not vital. As an example, we propose a similar circuit but with a PWM controlled inductor switching cycle. As said before, the switching power supply field is extensive enough that we could complicate it as much as we like.

It would be a great experience to be able to field test the drone and explore its usefulness on a real world application. For now, it is a step forward.

5.2 Promoting open-source

During this project we have used various open-source software and hardware, following this philosophy, the source code used for this application and schematics are available in Github at https://github.com/mastrain/Geiger-drone.git

Chapter 6: Bibliography

[1] E. Gallego, “Ministerio de defensa de España,” 5 2013. [Online]. Available: http://www.defensa.gob.es/ceseden/Galerias/destacados/publicaciones/docSegyDef/ficheros/053_LA_ENERGIA_NUCLEAR_DESPUES_DEL_ACCIDENTE_DE_FUKUSHIMA.pdf. [Accessed 15 1 2016].

[2] ArduPilot, “ArduPilot,” [Online]. Available: http://copter.ardupilot.com/. [Accessed 15 9 2015].

[3] DJI propulsion, “rcgroups,” [Online]. Available: http://www.rcgroups.com/forums/showthread.php?t=2046097&page=4. [Accessed 15 9 2015].

[4] A. Hand, “HobbyKing,” [Online]. Available: http://www.hobbyking.com/hobbyking/store/__66283__Gemfan_DJI_Style_Propeller_9_4x4_3_Locknuts_CW_CCW_2pcs_.html. [Accessed 15 1 2016].

[5] C. Anderson, “Do It Yourself Drones,” [Online]. Available: http://diydrones.com/. [Accessed 15 1 2015].

[6] Ardupilot, “Arducopter Radio Bandwidth,” [Online]. Available: http://copter.ardupilot.com/wiki/common-3dr-radio-version-2/#support_for_different_countriesregions. [Accessed 15 9 2015].

[7] L. Meier, D. Gagne and G. Grubba, “QGRoundControl,” [Online]. Available: http://qgroundcontrol.org/mavlink/start. [Accessed 20 2 2016].

[8] Git, “GIT,” [Online]. Available: https://git-scm.com/.

[9] Ardupilot, “Developer,” [Online]. Available: http://dev.ardupilot.com/wiki/building-ardupilot-with-arduino-windows/. [Accessed 23 2 2016].

[10] Ardupilot, “ArduPilot Development Site,” [Online]. Available: http://dev.ardupilot.com/. [Accessed 15 9 2015].

[11] Ardupilot, [Online]. Available: http://dev.ardupilot.com/wiki/learning-ardupilot-threading/. [Accessed 15 9 2015].

[12] G. F. Knoll, Radiation detection and measurement, John Wiley & Sons, Inc.

[13] Chrono, “Apollo NG,” [Online]. Available: https://apollo.open-resource.org/lab:pigi. [Accessed 15 1 2015].

[14] BroHogan, “DIY Geiger Counter,” [Online]. Available: https://sites.google.com/site/diygeigercounter/. [Accessed 15 1 2015].

[15] T. Napier, “Scribd,” [Online]. Available: http://es.scribd.com/doc/41802301/Nuts-Volts-25-01-Jan-2004#scribd. [Accessed 15 1 2015].

[16] J.-P. Charras, D. Hollenbeck and W. Stambaugh, “KiCad,” [Online]. Available: http://kicad-pcb.org/.

[17] Texas Instruments, “Texas Instruments,” [Online]. Available: http://www.ti.com/lit/ds/symlink/tlc555.pdf. [Accessed 15 1 2016].

[18] Arduino, “Arduino,” [Online]. Available: http://forum.arduino.cc/index.php?topic=236128.0. [Accessed 15 1 2016].

[19] M. Hämmerling, “Engbedded,” [Online]. Available: http://www.engbedded.com/fusecalc/. [Accessed 15 1 2016].

[20] A. Hand, “Hobbyking,” [Online]. Available: http://www.hobbyking.com/hobbyking/store/uploads/608361865X1450129X34.pdf. [Accessed 15 1 2016].

[21] Ardupilot, “APM:Copter,” [Online]. Available: http://copter.ardupilot.com/wiki/connecting-the-apm2/.

[22] APM, “Mission Planner,” [Online]. Available: http://planner.ardupilot.com/wiki/mission-planner-overview/#navigating_the_documentation. [Accessed 20 2 2016].

[23] Atmel, “Atmel,” Atmel, [Online]. Available: http://www.atmel.com/devices/atmega2560.aspx. [Accessed 20 2 2016].