Academic year: 2020-2021

LLSMF2018_ELEC

Technological and Quantitative Project

LOUVAIN

School of Management

Final rapport: Drone project

Professor: Macq Benoît

Assistants: El Khoury Karim, Samelson Jonathan, Sonneville Victorien

Group 7:

Deblander Nicolas 15661700 – Dercq Arthur 21321700 – Jolly Léopold 05641600 – t'Serstevens Géraud 37921700
Strome Guillaume 01201700

1. Project introduction

A. General context

In order to find a use for our drone, we have looked for a problem that could be solved efficiently with a drone. We have noticed that the average speed of road traffic in big cities is very low (for example: 11.3 km/h for the center of Brussels) which can be problematic in some situations. Furthermore, the speed limit in Brussels has come down to 30km/h and other big cities are following this trend. Of course, the ambulances are not impacted by this new legislation but nevertheless, we do not know the impacts that this change may have yet and the sizable congestions that it may cause. Thus, it seemed interesting to us to use the drone in some emergency situations because it is not impacted by the road traffic. After some reflection, we have decided that our drone would be a rescue drone being able to deliver medicines anywhere and quickly. Indeed, this drone can be very useful to assist firefighters, paramedics and the police but also people who need medicines urgently. For instance, the drone could bring insulin to a diabetic in the middle of an attack or a puff to an asthmatic in an asthma attack. It is following these different assumptions that we decided to carry out this project in this particular direction.

B. Designed solution

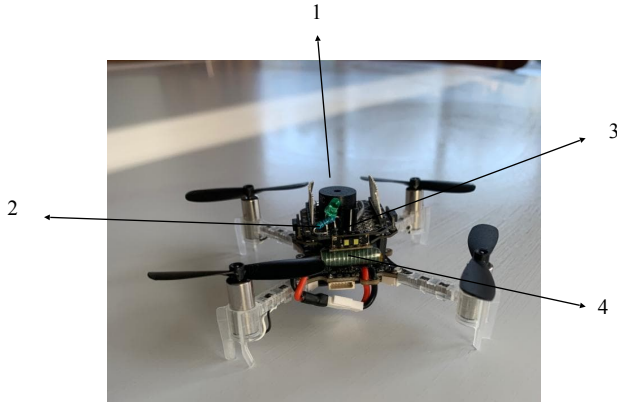
Our drone has the objective of delivering medicines anywhere and quickly. Thus, upon receipt of the order, the drone will leave the storage warehouse (starting position) to go to the customer's location. We assumed to our storage is supplied with the medicines needed in the scope of our project (insulin or puff for example). The drone will have to calculate the shortest route to get to the location and to avoid obstacles during its itinerary. After dropping off its package, the drone will return to the starting position, again, taking the shortest path.

In addition to this, we used a buzzer, a light as well as an infrared sensor. The light will stay on as long as the drone is flying. As for the buzzer, it will make noise each time our drone detects an obstacle. Finally, the infra-red sensor will have the task of checking if someone approaches to retrieve the medicine. Once the infrared sensor has detected that someone has approached, meaning retrieved the package, the drone will be able to take off and to return to the warehouse.

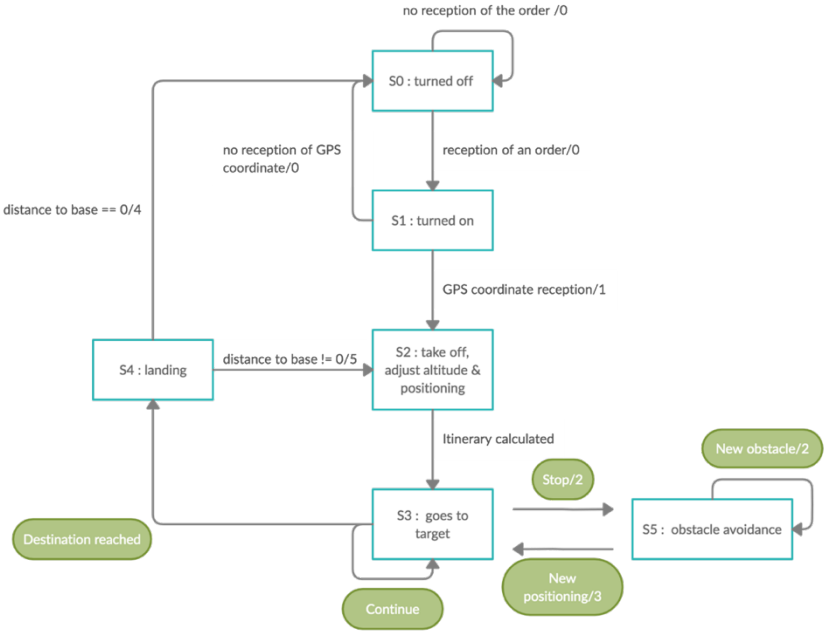
C. The components used

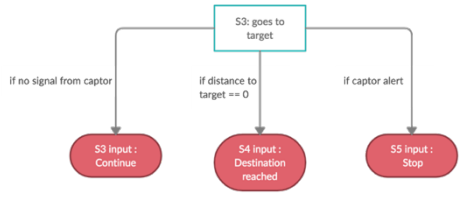
We have therefore used the following components to ensure that our drone is capable of achieving each of its objectives:

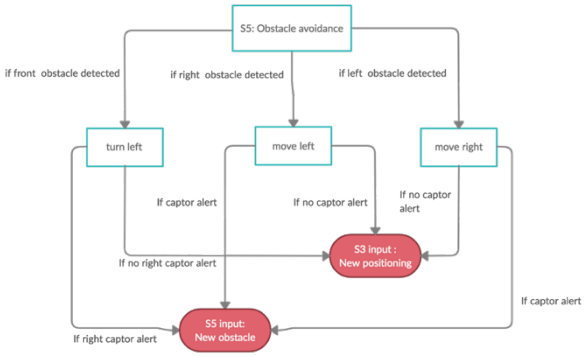
1. A buzzer
2. A green light
3. An infrared sensor
4. The different sensors of our drone which enable it to detect obstacles



A. General FSM





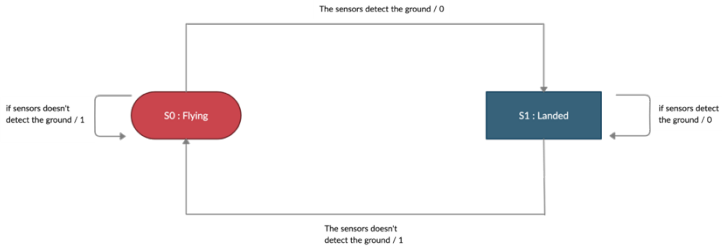


Our main FSM is therefore composed of 5 different states through which our drone will pass in order to complete its route. As we have not given the meaning of the outputs directly on the diagram, you can find them below:

- Output 0: The drone is not flying
- Output 1: The drone is flying
- Output 2: The drone must avoid an obstacle
- Output 3: The drone has avoided the obstacle
- Output 4: The drone has reached the base
- Output 5: The drone returns to base

In addition to our main FSM, we made a second FSM for the light system of our drone. This second FSM could easily have been integrated into the main FSM but we preferred to put it apart, for more clarity. This FSM shows that as long as our drone is flying, a light is on. The two possible outputs of this FSM are :

- Output 0 : light off
- Output 1 : light on



3. Explanation of our logic blocks

A. Go to target

Go to target is the third state of our FSM and aims at getting our drone to its destination. During this State 3, many things that we are going to see now are going to be calculated in order to allow our drone to reach its destination. The “go to target” state and therefore all the calculations associated with this state will have to be computed at the drone's departure and then each time our drone deviates from its initial trajectory due to an obstacle.

```
angle = abs(math.degrees(math.atan(x/y)))
```

- Calculation of coordinates

The x and y coordinates are the x and y values of the destination minus the current coordinates of the drone. We find the current coordinates of the drone using the StateEstimate function. This subtraction allows us to have the distances x and y values that remain to be covered. We will then be able to calculate the angle and the distance thanks to these coordinates.

```
x = x_destination - my_unsafe_logging_variable['stateEstimate.x']
y = y_destination - my_unsafe_logging_variable['stateEstimate.y']
```

- Calculation of the angle

The calculation of the angle will allow us to calculate the angle that our drone will have to turn to in order to find itself in front of the destination. It will then be able to move in a straight line to the destination. The value of the angle is calculated as the inverse of the

tangent. Subsequently, this calculation only gives us a value between 0 and 90 degrees. We then corrected this value to get the exact angle according to the quadrant.

- Calculation of the distance

The distance calculation allows us to calculate at any time how far our drone is from reaching its destination. This distance is computed using the Pythagore theorem.

```
distance=abs(math.sqrt(x**2+y**2))
```

B. Obstacle avoidance (S4)

First of all, we have created a definition called obstacle_detection(). This definition will use the multi-ranger package, which enables the drone's sensors to detect obstacles coming from any direction. Concretely, it allows our drone to detect obstacles in front of it, to its left and to its right. The definition will then return a value depending on whether it detects an obstacle to the left, to the right or in front of the drone.

Thanks to a second definition called is_close (range), we can choose a distance from which our drone will react to the obstacle. In our code, this distance has been set at 20 centimeters.

To be as accurate as possible, we have given precise instructions to our drone depending on the position of the obstacle. We are now going to cover each possibility.

- Obstacle Front

When a front obstacle is detected, we will use the following instruction:

```
while obstacle == 2:
    mc._set_vel_setpoint(0,0.1,0,0)
```

Consequently, the drone will be able to move laterally on the left as long as the obstacle is detected in front of him. The drone will move to the left thanks to the mc._set_vel_setpoint() function.
After avoiding the obstacle in front of him, we ask the drone to move another 15 centimeters to the left to make sure he has effectively avoided the obstacle.
The drone will then recalculate the angle to turn towards it and start moving in the direction of its destination (Transition state S3).

- Obstacle Left

If our drone detects an obstacle on its left, the drone will be instructed to move 20 cm to the right thanks to the function mc.move_distance().

```
if obstacle == 3:
    mc.move_distance(0.0,-0.2,0,velocity=0.1)
```

- Obstacle Right

The code to avoid an obstacle on the right of the drone is logically almost identical to the code to avoid the obstacle on the left of the drone. Here the drone will be instructed to move 20 centimeters to the left.

```
if obstacle == 4:
    mc.move_distance(0.0,0.2,0,velocity=0.1)
```

C. Light, Buzzer & Infrared Scanner

We also connected three additional features to our drone. A buzzer, an LED and an infrared reader. Our buzzer (pin IO4), just like a car, is activated when our drone detects an obstacle in its dodge zone and continues to buzz until this obstacle is avoided. As far as the LED (pin IO1) is concerned, it blinks when the drone is in flight. This allows it to be more visible. For the LED and the buzzer, the format (1) stands for on and the format (0) for off. As for the infrared reader (pin TX1), it is used to illustrate the fact that a customer is retrieving his order. When it returns a higher value than the critical value we've defined, the drone interprets that the customer has collected his order and can start its return process. We retrieve the IR values and activate the LED/buzzer with the following instruction:

```
cf.param.set_value('PROTO_LSM.IO1_ON','{:d}'.format(1))
```

```
cf.param.set_value('PROTO_LSM.IO4_ON','{:d}'.format(1))
```

```
IR=my_unsafe_logging_variable['Read_IR.getval_IR','float']
```

D. General definition

The last point of our code that we wanted to discuss with you is our definition called move_distance_to_target(). This last definition will take up almost all the definitions in our code and will be finally called in the main.

It is this definition that we sum up the steps of our general FSM.

After turning our drone in the right angle, allowing it to face its destination, we will use the following instruction:

```
while distance_restante>0.2:
    obstacle=obstacle_detection
    distance_restante(x,y)
    mc._set_vel_setpoint(0.1,0,0,0)
    if obstacle !=1:
        mc._set_vel_setpoint(0,0,0,0)
        time.sleep(0.1)
        obstacle_avoidance(obstacle_detection())
```

This means that as long as our drone is more than 20 centimeters from its destination, it will fly forward with a velocity equal to 0.1 m/s. In addition, if during its journey to its destination, our drone detects an obstacle (obstacle !=1), it will be able to avoid it thanks to the obstacle_avoidance() function. Once the drone is within 20 centimeters of its destination, it will be instructed to land. We took 20 centimeters to take into consideration the fact that

coordinates calculation is not 100% accurate, therefore, we wanted to prevent it from jmissing the destination.

4. Demonstration Briefing

In this section, we are going to discuss what will happen during our demonstration.

Firstly, we are going to enter coordinates that could hypothetically be the coordinates of a client. To stay in the limitation of the room, we will assume the client is located nearby.

As you might notice, we will not load the drone with a medication for some reasons that we will explain later in the drone's limitations section.

Afterward, our drone will then take off, calculate the distance and the angle to the target and start its journey.

In order to be as close as possible to the reality, it will face an obstacle on its way, which will be avoided without difficulty.

Once it has reached its destination, the drone will land smoothly.

On the ground, the drone will wait for the IR signal, equivalent to the successful reception of the order by the customer.

When the IR signal is received, the drone will be able to take off again, recalculate the angle in which it must position itself and the distance it must travel to return to its point of departure.

During this return, the drone will again encounter an obstacle to avoid.
Those two journeys will allow us to observe the use of the led and the buzzer too, as the led is on during the flight and the buzzer used during the obstacle avoidance.

5. Drone limitations

Our final project, although similar to what you have seen, is actually not exactly the same as our initial project. In this last part we will look at the difficulties we faced and how they impacted our initial plans.
We will distinguish three different types of limits: the limits related to the hardware, the limits related to our code and finally the limits related to project organization.

A. Hardware limits

Firstly, our main objective, as explained above, was to deliver medication. With the help of a flying object, the drone. Unfortunately, the size and the drone's strength doesn't allow us to

carry important load. In order to picture this delivery, our program prints "Order Delivered" once the drone has landed at its target and "Mission Accomplished" once back at the base.

To continue, after numerous attempts and a lot of broken propellers, we realized the drone was quite fragile. Thus, we decided to run tests only inside. Indeed, the drone is very sensitive to outside unexpected conditions such as strong wind or rain, and we wanted to avoid a too big collision with an external building, that would have put our project at risk.

As far as the components of our drone are concerned, we initially planned to use two LEDs. However, in order to explore more in details the potential of the hardware, we decided to swap these two LEDs for one led, a buzzer and an IR sensor. In fact, it has added some value to our code as well as to what the drone was capable of.

Finally, regarding our drone's battery, it doesn't have a very large capacity. Although we were careful to recharge our drone every time, we wanted to run the tests, it ran out of battery very quickly. We were able to make maximum three complete flights before the battery becomes flat. Therefore, we conclude that the drone could only fly a short distance to ensure that it doesn't break down in mid-flight.

B. Code limits

The main limitation we have been confronted to in our code is the StateEstimate function. We based our code on a drone functionality that normally returned us its position according to its initial flight position (i.e the base). However, in the last days of the project, this feature proved to be sometimes defective. Unfortunately, due to time constraints and because of the exam period, we didn't go beyond this limit. For that it would have been necessary to modify almost the integrality of our code, knowing that our main functions are based on this StateEstimate feature. However, we retain for the future, that using a physical formula based on the distance traveled as a function of speed and time of flight would have been a potential solution to this limit.

Then we initially planned to give our drone the possibility to scan a QR code to indicate that the customer had received his order. However, not having a camera nor the knowledge in python to implement a function to do such thing, we have modified this feature and changed it. We implemented an infrared reader that will return a value higher than a critical value when the customer will effectively have received his order. In fact, we were able to adapt and the IR function appears to be very effective.

Finally, regarding to obstacle avoidance, we focused on three types of obstacles to be taken separately, namely the obstacle in front, the obstacle on the right and on the left. We don’t want the drone to fly too close to buildings, this is the reason why left and right sensors are used. We therefore hypothesized that our drone wouldn’t face several obstacles simultaneously or wouldn’t end up in a dead end, which is actually quite justified as the drone should be able to fly at a high position. Dead ends or even L-shaped obstacles rarely

exist at such height, which makes us comfortable with this limitation, even though it doesn't take into consideration every possible obstacle.

C. Project limits

We observed a last limitation due to the current health situation. In our opinion, the work would have been more obvious in a present-day context than a distant one for several reasons. First of all, we sometimes lost a lot of time to carry out certain steps due to computer problems that could have been solved much more quickly in a face-to-face context (e.g., problems related to the installation of cfclient and cflib in the terminal). The bi-monthly project meetings would, in our opinion, have been more productive if they had been held in person. Being beginners in python programming, it was sometimes difficult for us to express our questions about the code or to explain the actions performed by our drone in Teams calls. Finally, in our opinion, this group project is a project where it is important to be able to see each other face-to-face in order to, for example, carry out tests with the drone, which was less obvious given the health situation (lockdown at home, mandatory quarantine, etc.).

6. Conclusion

In conclusion, this work allowed us to learn how to manipulate and create definitions under the python software. It also taught us the importance of working methodically and running multiple tests in an object-oriented programming language, in order to more easily observe the roots of a problem.

Despite of some limitations mentioned before, the project is theoretically able to fulfill its promises.

If hypothetically we could overlook the power, battery and range limits of the drone, it should be able to travel loaded with medication up to a target, land, detect that the customer has recovered his order before taking off again and return to its starting point.

During this journey, the drone blinks a LED in order to be more conspicuous for its surroundings by warning them that it is in movement. In the same idea as a car, the drone emits a sound signal using its buzzer once it detects an obstacle in its critical area. It will then avoid the obstacle before repositioning itself and resuming its course to its objective.