Mapping of unknown environments with an autonomous drone swarm

2019 Group 11-17

Tan Kai Cong (4O124)
Lee Yue Heng (4O111)
Tai Zi Xiang (4O122)

Hwa Chong Institution (High School)

Abstract

Mapping is the act of creating a model of an unknown environment, usually through systematic navigation and observation. Mapping using fully autonomous robots, even without absolute positioning systems such as GPS, can be accomplished through Simultaneous Localisation and Mapping (SLAM) methods, which accumulates onboard sensor data over time to generate information about both the robot's position and the surrounding environment simultaneously. However, majority of SLAM research so far utilise Unmanned Ground Vehicles, which limits the robot's ability to create a full 3D model of the environment. In addition, most SLAM research use sensors such as cameras or laser range-finders that are bulky or costly. Hence, this project presents a unique approach that utilises multiple cheap, micro-sized drones in conjunction with an appropriate SLAM algorithm that allows them to operate as a swarm-based intelligence. To achieve this, fully customised drones were created using 3D-printed frames, PCBs and minimal sensors. A simplified Hector SLAM algorithm was implemented to allow the drones to gather environmental data as an autonomous swarm, which was transmitted wirelessly through Radio Frequency modules to a laptop that translated the data into a 3D map. The solution was tested in multiple scenarios and it was found that the drones were able to produce maps of decent quality. The time taken for mapping completion was also greatly reduced when more drones were used. The solution design is versatile and can be implemented in almost any real-life scenario requiring information of an environment that is inaccessible or unsafe for humans, such as mine mapping, search and rescue operations and even gathering military intelligence.

I. Introduction

In recent years, 3D robotic mapping has gained a lot of interest in many engineering related industries, especially with the advent of Unmanned Aerial Vehicles (UAV) that can easily navigate to areas that are inaccessible to traditional ground vehicles (Shang & Shen, 2018). Some companies have developed commercially available products around this field. Emesent, a drone autonomy and data analytics company launched its first commercial product, Hovermap, in November 2018. It is a payload that can be attached to any drone and provides autonomous indoor and outdoor mapping capabilities for the drone. Their product has seen extensive use in mining, building inspections and construction monitoring. It works by using a LIDAR, a detection system which uses laser as a RADAR, and Simultaneous Localisation and Mapping (SLAM) based mapping (Emesent, 2019).

SLAM is an advanced technique for a mobile robot to build a map of its environment while simultaneously estimating its position on the map. Majority of research so far regarding SLAM on UAV has been focussed on Visual SLAM, where a camera is used as the predominant sensor for gathering environment data (Artieda et al., 2009). Using optical sensors can be beneficial as compared to other sensors typically used like LIDAR, RADAR and ultrasound because of its ability to provide rich visual data while being relatively lightweight and cheap. However, Visual SLAM does have its drawbacks. A relatively powerful onboard processor is needed in order to process the images at a high FPS (frames per second) in real time since the algorithms used for Visual SLAM and map reconstruction are computationally intensive (Nex and Remondino, 2014). Also, while optical sensors are cheaper and easier to install as compared to traditional depth-based sensors, they still cost a few hundred dollars or are rather bulky.

A novel SLAM approach that focused on creating multiple small, cheap and simple drones that could operate as a swarm and used basic SLAM algorithms, rather than developing a single drone with expensive components and complex algorithms, could be a solution to some of the limitations of current mapping methods. Using such an approach has the potential of reducing scan times. No matter how advanced a single drone can be, there will be some upper limit as to how fast it can create a map without sacrificing too much on map quality. With a drone swarm, increasing the number of drones used can, in theory, linearly reduce the amount of time needed to complete a scan, while maintaining the same quality. Another benefit would be versatility. Each individual drone will be quite small, given its simplicity, and thus can be utilised in a wide variety of environments, even cramped places. In contrast, a single large-sized and complicated drone would require extreme precision in control in order to navigate space-constrained locations. Also, swarm-based robots can serve to provide economic benefits. Each individual drone is easy to manufacture and thus can be mass-produced cheaply (Scharre & Marshall, 2014). This also reduces losses in the case of malfunction as an individual drone in a swarm is less valuable than a single complex drone.

This research attempted to create this novel SLAM approach.

II. Solution Design

The solution implemented had 5 main aspects, each with its own unique features that allowed for a comprehensive solution design. Figure 1 shows the final assembled drone.

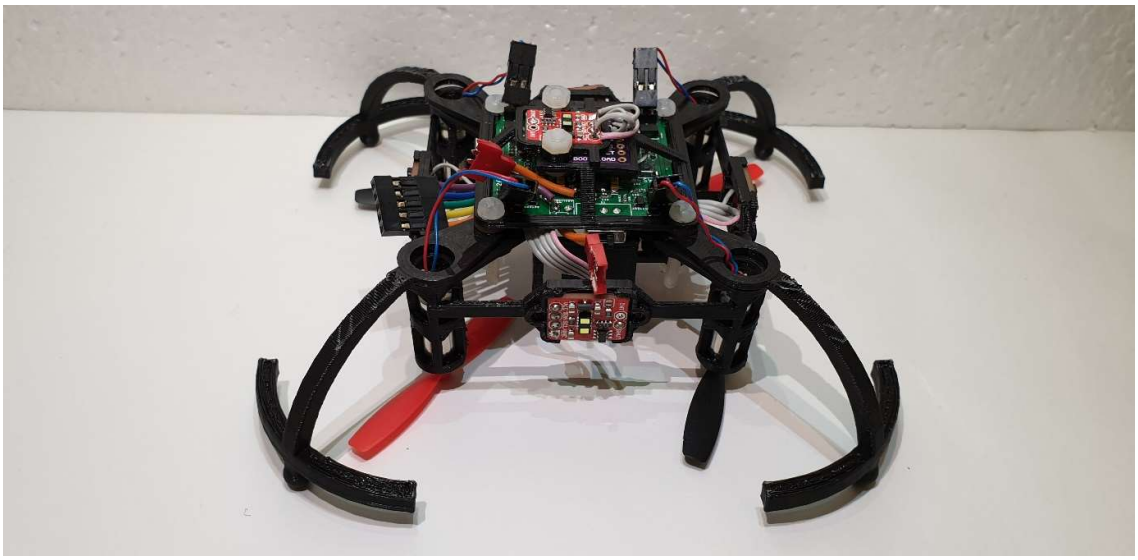


Figure 1: Completed drone assembly

A. Mechanical Design

The 3D model of the drone was designed using Autodesk Fusion 360. There were 4 main sections in the drone's design: (Fig 2)

- Main frame: A contiguous 3D printed structure holding majority of the drone's components
- Landing Gears: 4 separate pieces that elevated the drone off the ground slightly while covering the propellers, protecting them
- Printed Circuit Board: Held all the electrical components and circuitry
- Top frame: A minimally designed holder that housed the IMU and top ToF sensor

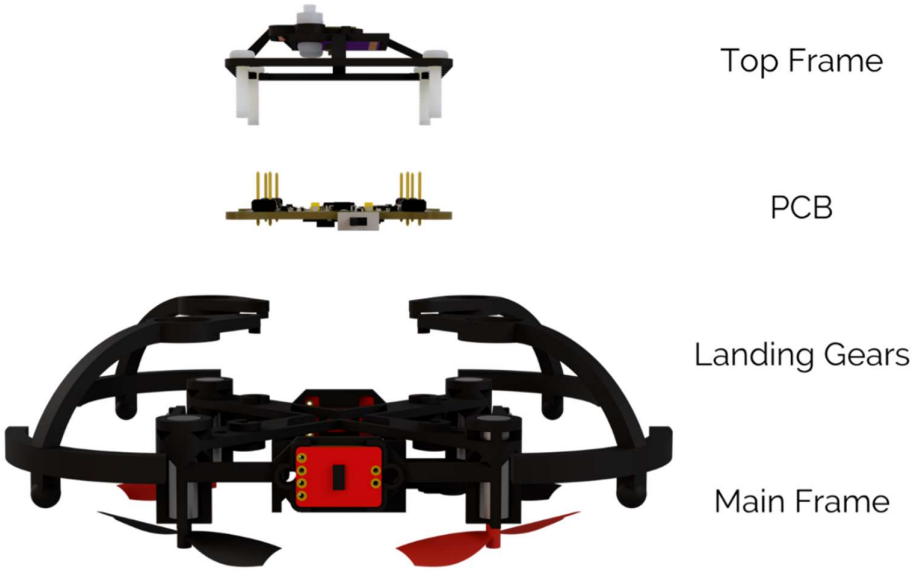
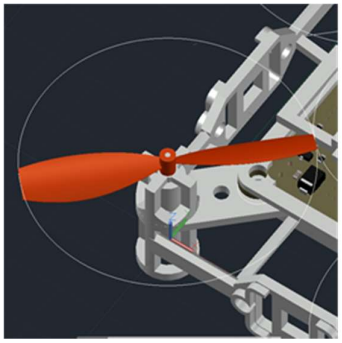


Figure 2: Exploded view of the drone

The drone also had several important engineering considerations:

1) Inverted Motors: The drone was designed with inverted motors due to the extremely compact design (Fig 3). In the first prototype with the motors pointing upwards, it was found that the propeller was blowing down onto the drone's body itself almost 20% of its area of travel (Fig 4). This translated into each motor providing only 60% of its available thrust, which was extremely inefficient. Inverting the motors solved this problem while maintaining the same small form factor.

Figure 3: Inverted motors



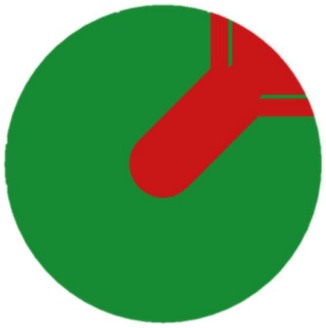


Figure 4: Upright motors (left); Area of propeller travel (right)

2) Precise component holders: Since the frames were all 3D printed, all the component holders, namely the motors, battery, ToF sensors and IMU were precision engineered to fit these components snugly, such that they could be fixed in place perfectly with no additional clamping or gluing required (Fig 5). This allowed for quick assembly as well as accurate placement of sensors.

Figure 5: ToF sensor holder

3) Detachable propeller covers: The propeller covers protected both the user as well as the propellers themselves. These covers also function as landing gears, elevating the drone 5mm above the ground. The cover was intentionally made with a small amount of slack such that in the event of a crash or malfunction, it could absorb some of the impact, reducing the chance of breakage. Even in the worst-case scenario of breaking, since the cover was detachable, it could be easily replaced.

4) Efficient frame design: During the design process, adding holes and thinning of certain sections were done as much as possible without sacrificing too much on structural integrity. This resulted in an extremely efficient frame design that only weighed 13g despite holding basically all the components together. In addition, all the 4 sections of the drone were held together by only 4 screws on each corner, which allowed for quick and simple assembly.

B. Electrical Design

A Printed Circuit Board (PCB) was created to hold all the relevant components. The PCB was designed using Autodesk EAGLE, which allowed the full use of Eagle-Fusion interoperability, since the PCB designed in EAGLE could be easily pushed to Fusion 360 as a 3D model for continual updating of the drone design (Fig 6). The design was sent for manufacture by JLCPCB, a Chinese PCB fabrication house.

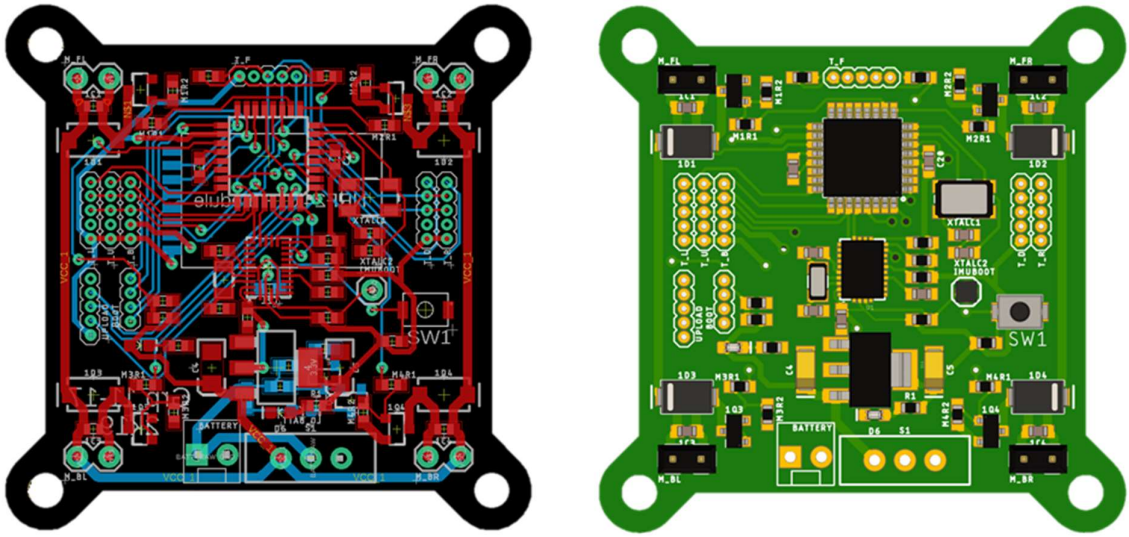


Figure 6: PCB design on EAGLE (left); 3D model export on Fusion 360 (right)

The PCB had an extremely small form factor of only 44×44mm by heavily utilising surface mount (SMD) components. (Fig 7) The board consisted of 4 MOSFET circuits for motor control, an SMD breakout for the RF module, through-hole pads for sensor connections, a 3.3V onboard voltage regulator (LDO), and a microcontroller (MCU) chip. The MCU used was the ATMEGA328P, with its circuitry based off the 3.3V Arduino Pro Mini. The appropriate Arduino bootloader was also burned onto the MCU using an Arduino Due, which allowed the MCU to be programmed from the Arduino IDE. A handmade circuit board was created to easily connect the Arduino to the PCB for programming. (Fig 8)

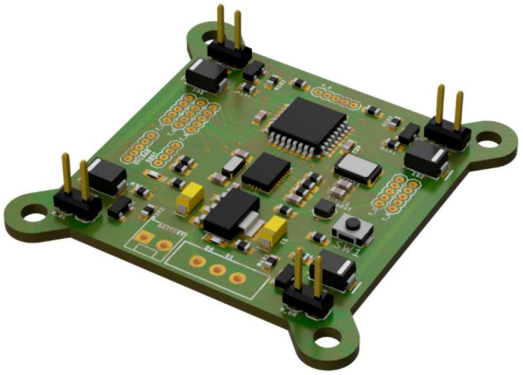


Figure 7: SMD components on PCB

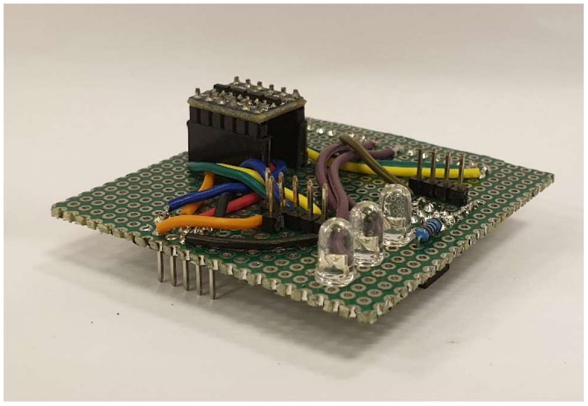


Figure 8: Programmer board for Arduino

To ensure the safety of the 1S Li-Po batteries used, an independent low voltage LED indicator circuit using 2 NPN transistors was incorporated into the PCB. (Fig 9) Since Li-Po cells have a nominal voltage of 3.7V, the resistor values were configured so that the LED would start lighting up at 3.7V and would get brighter as the voltage decreases from 3.7V. In addition, a voltage divider output was connected to the MCU's analog input pins, which allowed it to calculate the battery's exact voltage and automatically land the drone when the battery voltage went below 3.4V. (Fig 10)

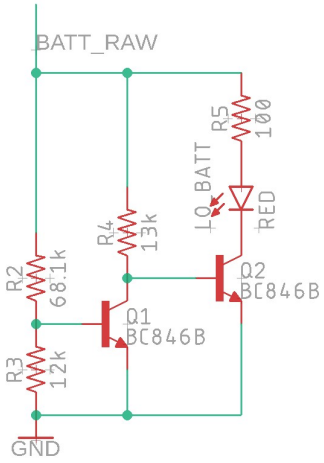


Figure 9: Low voltage indicator

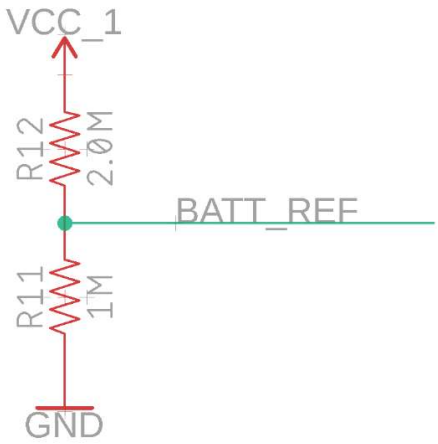


Figure 10: Voltage divider connected to MCU

However, a major problem faced was that the original PCB design intended to incorporate the BNO080 IMU as a tiny SMD chip (only 5.2×3.8mm), but the chip had an LGA-28 pad layout, which meant the solder pads were located directly underneath the chip, making it very difficult to solder and near impossible to debug. (Fig 11) Since the chip was not able to work, an older model, the BNO055, was utilised together with a commercially available Chinese breakout board instead. Despite its bigger size, this was seamlessly integrated with the top frame. (Fig 12)

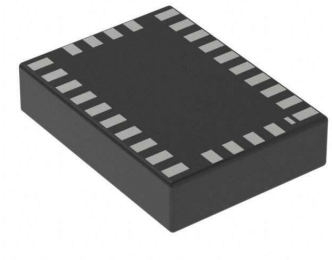


Figure 11: BNO080 chip

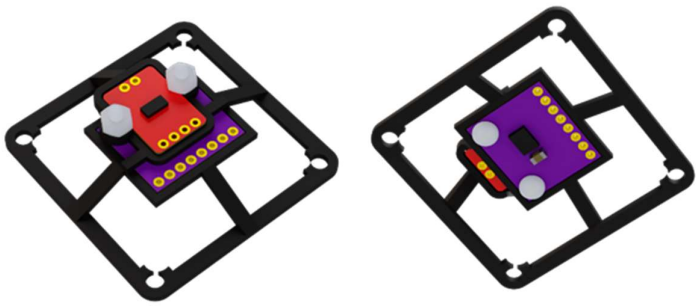


Figure 12: Top frame top view (left) and bottom view (right)

Another problem faced was the motors drawing too much current as it worked. When the motors drew high currents, the battery voltage dropped drastically. This meant the LDO ended up outputting less than the 3.3V needed for the MCU to control the MOSFET circuits that ran the motors. To fix this, 2 smaller capacity batteries were used. One battery was still connected as usual to power the motors. However, the voltage input to the LDO was isolated from the PCB and an additional battery was connected in series to it (Fig 13 & 14). This meant the LDO is always supplied with at least 7V, which it steps down to a stable 3.3V output, while the motors were able to operate within specifications from one battery only.

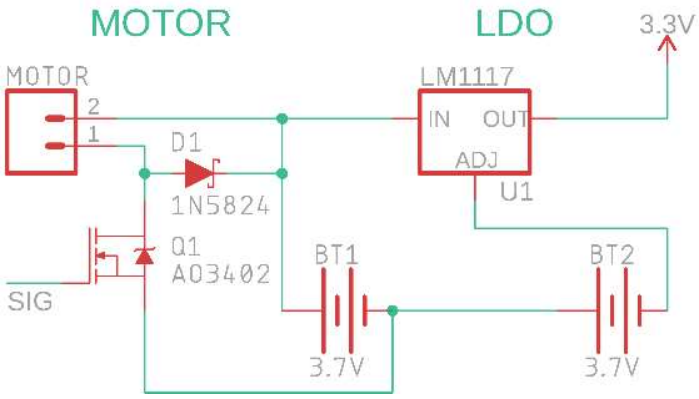


Figure 13: Simplified power supply schematic

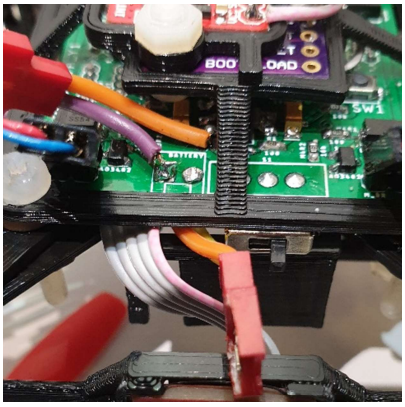


Figure 14: Modified battery connectors

C. SLAM Algorithm

SLAM stands for Simultaneous Localisation and Mapping and is a challenging problem as the robot needs to generate a map without prior knowledge about its location, but the need to know its location is crucial for mapping to be done accurately. The standard procedure for solving SLAM problems is as listed by Mangharam & Jain (2016) (Fig 15):

- An initial scan of the environment is taken and registered as an initial map, with the starting position of the robot taken to be the origin of the map.
- The robot changes its position by some amount and a new scan is taken. Based on the position change of certain landmarks in the new scan, the transformation of the robot (i.e. the robot's location) can be estimated.
- Align the new scan to the robot's position estimate and update the map.
- This process of moving, estimating position change and updating the map is repeated until a complete map is generated

SLAM algorithms vary based on path planning, the method used for map creation, how the position change is estimated, etc.

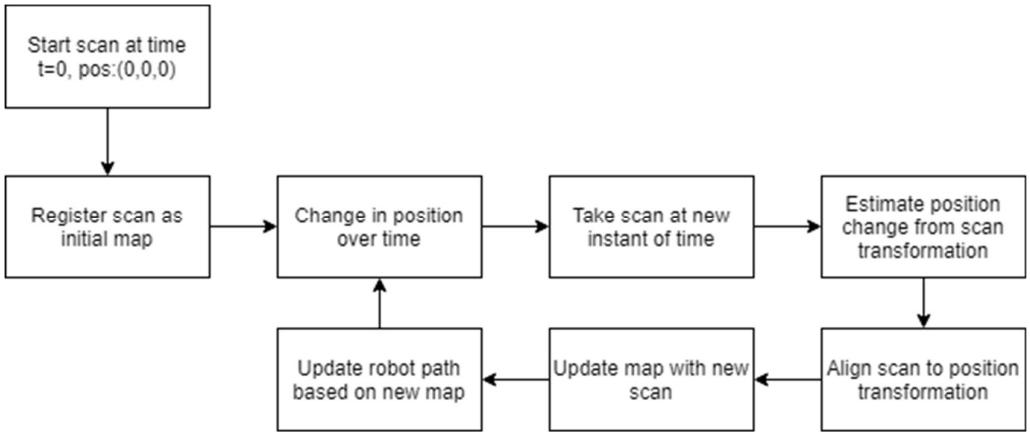


Figure 15: Typical SLAM approach flowchart

The SLAM method used in this research was a simplified version of Hector SLAM. Hector SLAM is a unique SLAM approach that does not require odometry (motion sensor data) and regards the robot as a 6 Degree of Freedom (6DOF) rigid body, rather than a 3DOF robot that most traditional SLAM approaches take. Hence, this method could generate more accurate 3D maps as it considered both the translational motion and rotational motion of the sensors.

Hector SLAM uses Occupancy Grid Mapping, which represents the map as a grid of cells that have a value between 1 to -1, which are logarithmic probabilities of them being Occupied, Unexplored or Free. During the scan matching, the new scan is matched to the previous scan through an iterative process that calculates the change in position of arrays of location points using a set of complex functions. The transformational and rotational matrix of the transformed new scan is used to estimate the pose change of the robot and added onto the Occupancy Grid map. In addition, this SLAM method generates multi-resolution maps using different grid sizes, with coarser grids being used for quick path planning calculations and the basis for calculating pose changes in finer grids. (Kohlbrecher, Stryk, Meyer & Klingauf, 2011)

Figure 16 and 17 shows a simplified diagram of how this mapping works. In the real world, the robot is represented by a blue square and the cone is the scanning range of the robot. On the grid map, Unexplored cells have a value of 0 and are red, Occupied cells have a value of 1 and are grey, Free cells have a value of -1 and are white.

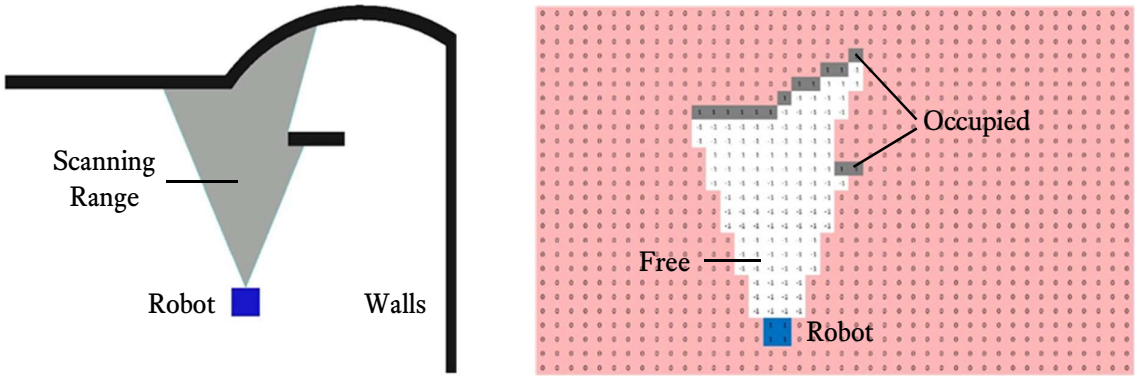


Figure 16: Actual world (left) vs Grid map (right) at initial pose

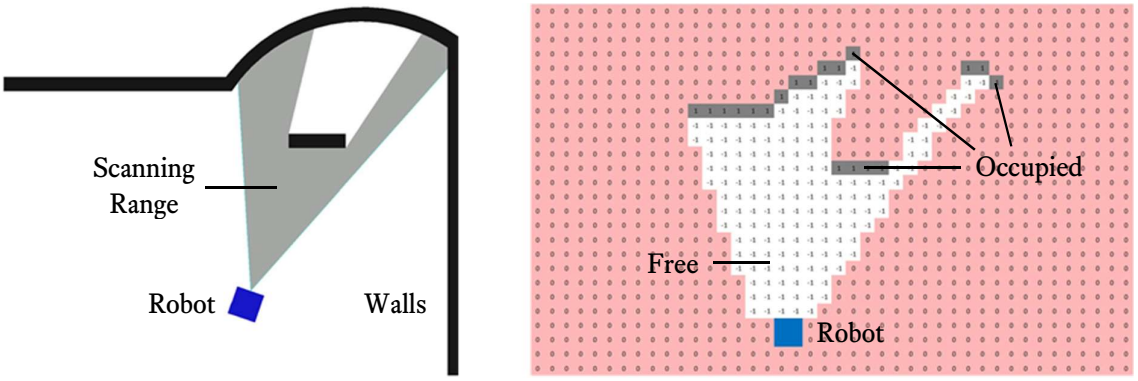


Figure 17: Actual world (left) vs Grid map (right) after robot rotation

The robot's path is based on the concept of frontiers, which are simply the boundaries between Free and Unexplored cells in the map grid (Fig 18). The drone simply moves to the closest frontier to itself repeatedly until there are no more frontiers remaining.

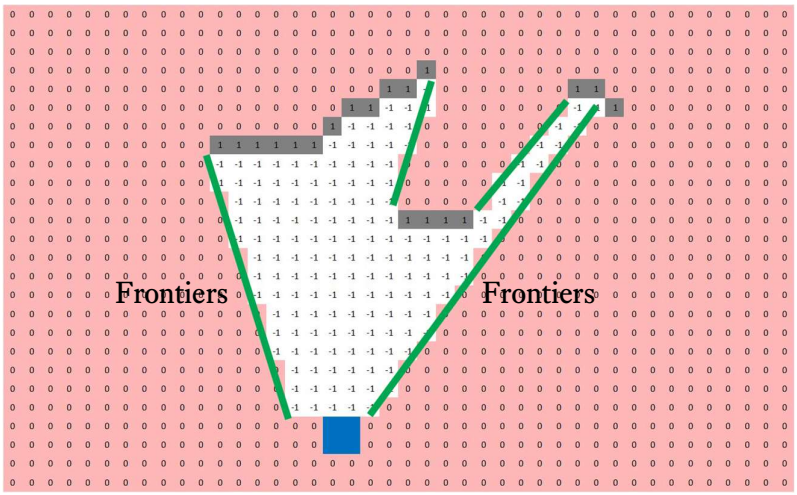


Figure 18: Green lines show the map frontiers

This complex SLAM method was implemented using the Robot Operating System (ROS) package, *hector_mapping,* since this package condensed the calculations into relevant functions. This package was simplified and ported over to Arduino for use as a programming library.

The drone used 1 IMU and 6 ToF (Time-of-Flight) sensors to do the mapping. There was one ToF sensor on each side, one facing up and one facing down. The ToF sensors were highly accurate distance sensors that worked by sending out a short laser pulse and taking the time between sending and receiving to calculate the distance of objects. These 6 ToF sensors provided the drone with depth-based information about the environment which was used to generate the map. During movement, the drone used the IMU to control its motors to maintain a constant angular velocity about the Z axis. This simulated the function of a LIDAR and provided the drone with a full 360° view of the environment.

D. Data Processing

The MCU that the drones used had very limited memory. Hence, all the memory-intensive map generating processes was done offboard on a more powerful processor (in this case, a laptop). Each drone had an RF module that transmitted the 3D occupancy grid data as well as its positional information to another RF module connected to the laptop (Fig 19). The data received was streamed into a program that parsed the data and recorded the map information and the drone's position relative to the universal map frame. This data could be transmitted to the drone when requested.

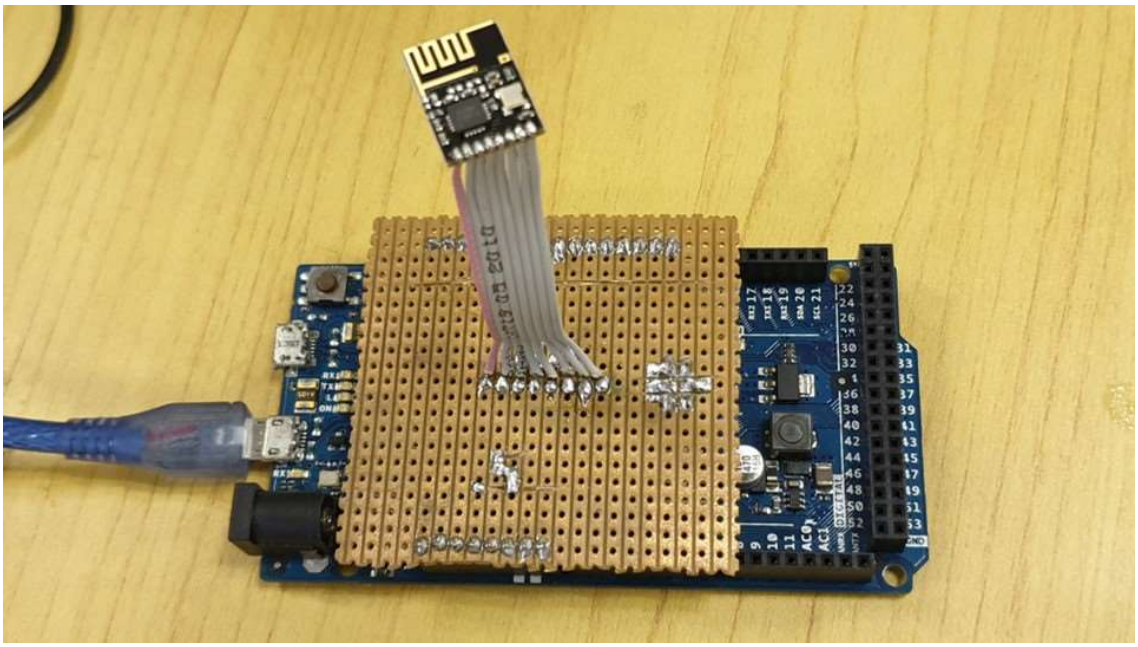


Figure 19: RF module circuit board for Arduino

At the end of the mapping process, the occupancy grid coordinates could be exported as a text file and imported into a point cloud processing software to view the generated map.

E. Swarm-based Implementation

The aspects of the solution design had been specifically optimised for implementing into a swarm system. Assembly of the drone was simple and given the standardised design, calibration of sensors and movement could also be done relatively quickly. The RF module could connect with up to 6 other modules, and could even support tree topology networks, with a theoretical maximum of 3125 modules.

Coordinating a swarm system for this problem did not actually require a highly sophisticated communication protocol. By assuming that all the drones were placed close to one another initially, the swarm implementation could be simplified greatly. The first drone's position was taken to be the origin of the map and after it had travelled some distance away, the second drone was activated. The second drone's initial scan was compared with the first drone's initial scan to determine the second's drone's position and this data was added onto the map. The second drone would now travel to the frontier closest to itself that was also outside of a certain radius of the first drone. Since the map generating was already done on an offboard, common control point, any additional drone could simply repeat this process and add

onto the map data, while naturally avoiding collisions by staying out of the way of other drones when finding frontiers.

This implementation preserved the drone's independence of operation since it was not directly reliant on any other drone for its proper function. This made it extremely easy to scale up or down the number of drones used and would ensure the mapping was completed even if certain drones malfunctioned.

III. Results and Discussion

A. Results

With the conclusion of a scanning sequence, the coordinates of the Occupied cells from the Occupancy Grid was exported to generate a point cloud of the environment. The software the team used to generate the maps was *CloudCompare*, an open source point cloud processing software. Doing post-processing allowed the map to be improved in quality by cleaning it up using noise filters and interpolating new coordinates to produce a fuller map. The points could also be used to create a mesh (3D surfaces) of the environment.

Scanning parameters were varied and the results were compared. By halving the grid size used, there was a minor time reduction but a visible drop in map accuracy. This was likely due to the drones' paths being basically the same even with lower grid resolutions. However, using two drones instead of one approximately halved the scanning time while maintaining the same map quality, since the drones conducted mapping simultaneously.

Figures 21 and 22 are maps of Figure 20. Figure 21 is a raw point cloud, Figure 22 is a map where the walls had been cleaned up with filters and generated into a mesh.

Figures 24 to 27 are maps of Figure 23. Figure 24 and 25 are point clouds and meshes, respectively, that were created using an occupancy grid size of 20cm. Figure 26 and 27 were created with a larger grid size of 40cm. Comparing the meshes of Figure 25 and 27, it is clear that a larger grid size does have a visible effect on the map accuracy.

Figure 20: Cube shaped environment with many cupboards and boxes

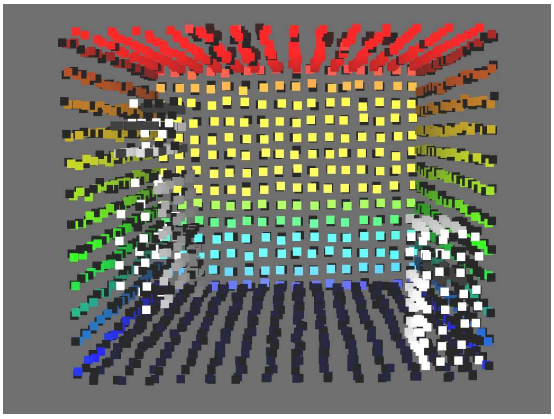


Figure 21: Point cloud with grid size 20cm

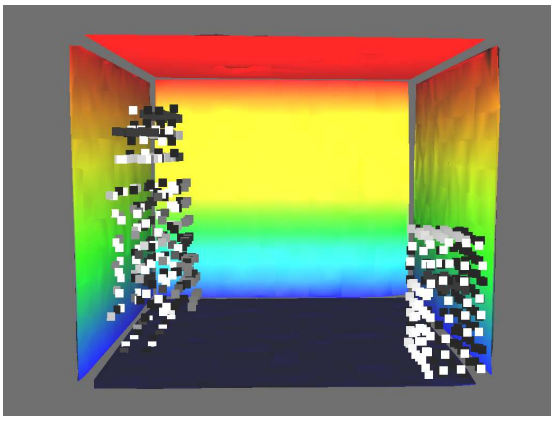


Figure 22: Partial mesh generated from 20cm grid

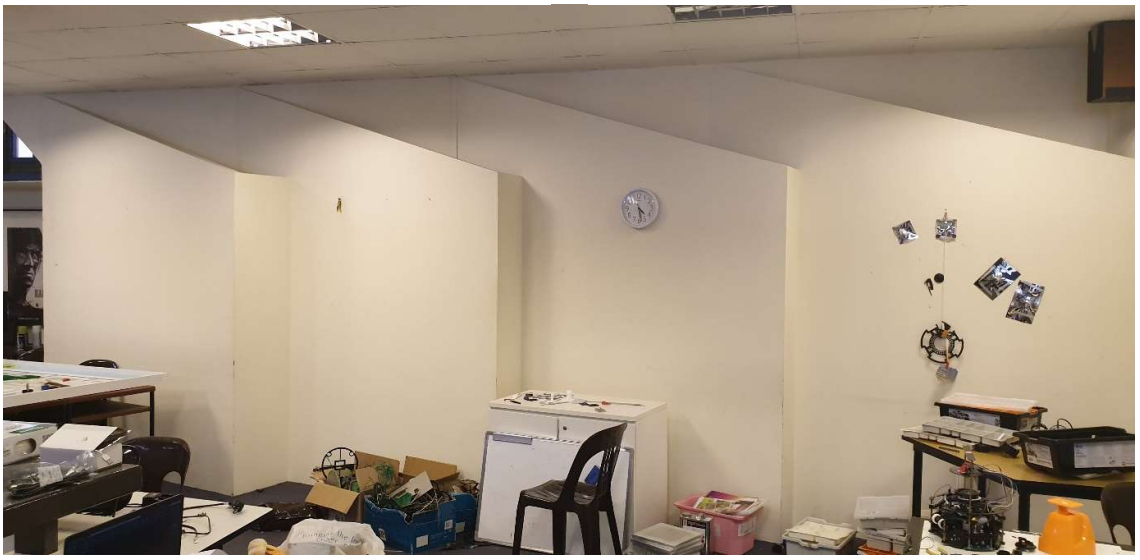


Figure 23: Slanted wall with many tables

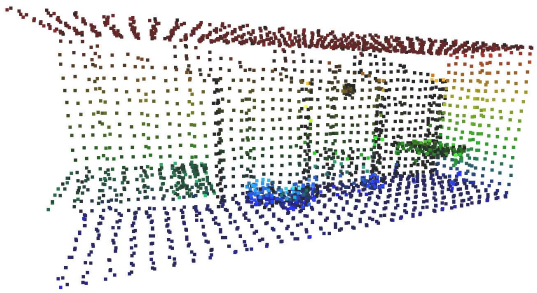


Figure 24: Point cloud with grid size 20cm

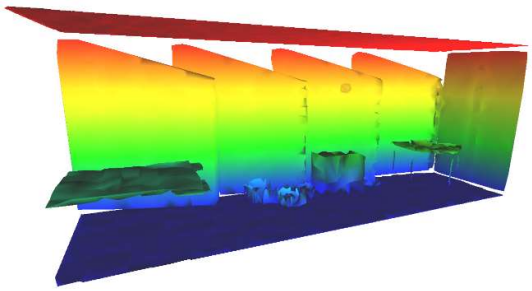


Figure 25: Mesh generated from 20cm grid

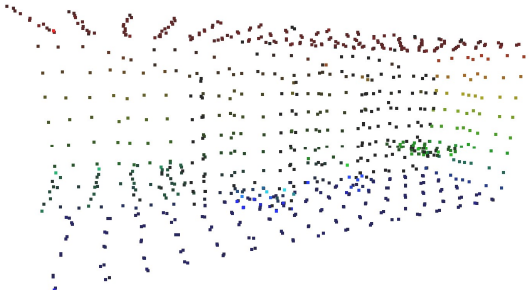


Figure 26: Point cloud with grid size 40cm

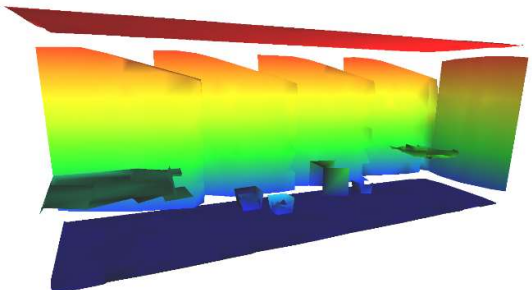


Figure 27: Mesh generated from 40cm grid

B. Limitations

There were some limitations during experimentation. Due to lack of resources, only two drones could be assembled for testing. While addition of drones is theoretically easy to implement, stability of larger swarms such as latency tests could not be done since only two drones were made. Time reduction gains with larger swarms could not be properly calculated as well.

Also, each drone had an average run time of only 6 minutes (average continuous current consumption of 3A with battery capacity of 300mAh). This made it impossible to test mapping on larger rooms since the drones could not fly long enough to complete a full map of the environment.

During the post-scanning processing, where the map was cleaned up using certain filtering and interpolation techniques, there might have been some bias in the tuning since the actual environment was already known. Hence, the results shown might be slightly cleaner than what an actual raw scan might have look liked.

C. Future Improvements

A main improvement would be to increase the drone's runtime. This can be done by switching from the 7mm coreless motors currently used to the larger 8.5mm motors that can generate more thrust while keeping almost the same small size, which allows the drone to carry heavier, higher capacity batteries.

Another possible improvement would be to use a more powerful MCU. A faster MCU would be able to perform the complex mapping functions at a higher frequency, which should be able to generate more accurate maps using a smaller grid size. An MCU with more pins could allow for easier debugging by connecting more programmable indicator LEDs to the extra pins.

A software improvement would be to explore the use of OctoMaps which is a special way of map representation that uses Oct-Trees instead of conventional grids. Using Oct-Trees significantly decreases the amount of data needed to represent a map as demonstrated by Musba (2013), which decreases the amount of latency during data transfer. In fact, if a more sophisticated communication network is used, a decentralised system (without sending data to laptop) could be established for increased autonomy.

IV. Conclusion

A highly versatile and scalable method for autonomous 3D mapping using micro-drone swarms was explored in this research. A small, cheap and easily assembled drone could be made using technologies like 3D printing and PCBs. The drones' ability to operate as a swarm-based intelligence, generating readable 3D maps of an unknown environment with SLAM algorithms was also successfully demonstrated.

This method presented is mainly suitable for mapping situations where having quick scan times are more important than having extremely high accuracy. Scaling up the size of the swarm is straightforward since the drones are not directly interdependent on one another; which makes decreasing the scan time very easy. This method also excels in environments where space is constrained or if the point of entry is very small due to the small size of each individual drone.

Hence, this solution is highly applicable to a wide variety of scenarios. Some possibilities include time-stressed search and rescue missions, exploring human-inaccessible underground mines, monitoring performance on construction sites, gathering intelligence of a hostile location; the possibilities are endless, and it would be exciting to see more developments and applications in this field of robotics in the near future.

V. Acknowledgements

The authors would like to express their sincerest appreciation for their mentor, Mrs Ng, for her invaluable guidance throughout this project.

VI. References

Artieda, J., Sebastian, J. M., Campoy, P., Correa, J. F., Mondragón, I. F., Martínez, C., & Olivares, M. (2009). Visual 3-D SLAM from UAVs. *Journal of Intelligent and Robotic Systems,55*(4-5), 299-321. doi:10.1007/s10846-008-9304-8

Digi-Key Electronics. (2017). 28 Pin LAG [Digital Image]. Retrieved from https://media.digikey.com/Renders/Hillscrest Labs/28-Pin-LAG_sml.jpg

Emesent. (2019). Hovermap. Retrieved from https://emesent.io/products/hovermap/

Girardeau-Montaut, D. (2019, February 24). CloudCompare (Version 2.10.2) [Computer software]. Retrieved from http://www.cloudcompare.org/

Hess, W., Kohler, D., Rapp, H., & Andor, D. (2016). Real-time loop closure in 2D LIDAR SLAM. *2016 IEEE International Conference on Robotics and Automation (ICRA)*. doi:10.1109/icra.2016.7487258

Kohlbrecher, S. & Meyer, J. (2018) Hector_SLAM (Version 0.3.5) [Source Code] Retrieved from https://github.com/tu-darmstadt-ros-pkg/hector_slam

Kohlbrecher, S., Stryk, O. V., Meyer, J., & Klingauf, U. (2011). A flexible and scalable SLAM system with full 3D motion estimation. *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*. doi:10.1109/ssrr.2011.6106777

Lachow, I. (2017). The upside and downside of swarming drones. *Bulletin of the Atomic Scientists,73*(2), 96-101. doi:10.1080/00963402.2017.1290879

Mangharam, R. (Producer), & Jain, P. (Writer). (2016, April 27). *Lecture 3 2: Hector Mapping - Simultaneous Localization and Mapping* [Video file]. Retrieved July 28, 2019, from https://www.youtube.com/watch?v=Q4qM-Uzj1SI

Musba, I. (Producer). (2013, September 10). *3D Mapping and Navigation using Octrees on a Quadrotor* [Video file]. Retrieved July 28, 2019, from https://www.youtube.com/watch?v=7epTZM0yHA4

Nex, F., & Remondino, F. (2013). UAV for 3D mapping applications: A review. *Applied Geomatics,6*(1), 1-15. doi:10.1007/s12518-013-0120-x

Scharre, P., & Marshall, J. (2014). The Coming Swarm: The Quality of Quantity. *Robotics on the Battlefield,* 2nd ser. Retrieved from https://s3.amazonaws.com/files.cnas.org/documents/CNAS_QualityofQuantity_Scharre.pdf?mtime=20160906081909.

Shang, Z., & Shen, Z. (2018). Real-Time 3D Reconstruction on Construction Site Using Visual SLAM and UAV. *Construction Research Congress 2018*. doi:10.1061/9780784481264.030