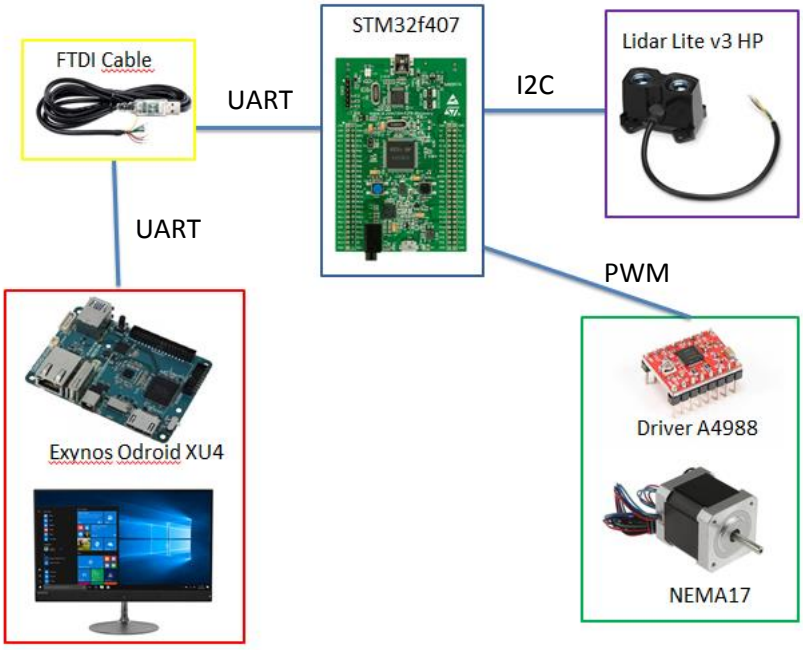
Final Report Project GSE5 2018/19

BIN IBRAHIM Ahmad Hanif Imran
BIN SHAHIDAN Muhammad Imran

I. INTRODUCTION

The objective of this project is to design a real-time navigation system for a robot vehicle. So, the acquisition device is based on a portable autonomous laser scanner called Lidar which is controlled by a STM32f407 and an Exynos board that can both communicate through a serial/USB connection. In our case, the Lidar data will be sent from STM32f407 to the PC base station running simultaneous localization and mapping (SLAM) and visualization software by using a serial connection (FTDI Cable). The acquisition device and the base station will communicate by using Robot Operating System (ROS) and a WIFI connection. The SLAM algorithm is based on Google Cartographer and a 3D visualization tool called RVIZ, both integrated in ROS. So, we can see the functional schematic of the project in the figure below:



We have divided our functional schematic to five parts :

- **STM32f407** : to control the Lidar and the NEMA17 stepper motor
- **LIDAR** : to measure a distance to a target
- **FTDI Cable** : a serial connection in order to send Lidar data from STM32f407 to Exynos board
- **Driver A4988 and NEMA17**: to rotate the Lidar in order to get the Lidar data from all the sides
- **Exynos Odroid XU4 and desktop** : PC based station which is running SLAM and visualization software.

II. CONCEPTION

a. System

Ubuntu 18.04 for Odroid

The ODROID-XU4 is basically a heterogeneous multi-processing Octa-core Linux Computer and supports the Linux Kernel 4.14 LTS. Offering open source support, the board can run various flavours of Linux, and in our case, we're using the latest Ubuntu 18.04 and the latest versions of Android. So, we need to install Ubuntu 18.04 on a 32GB micro SD card for the Exynos board and make sure to use a GCC 5.4 compiler. Besides that, we use Robot Operating System and Google Cartographer to make the mapping.

b. Materiel

STM32f407

The STM32F407xx family is based on the high-performance ARM®Cortex®-M4 32-bit RISC core operating at a frequency of up to 168 MHz. So, basically we're using this STM32f407 as the platform between Lidar and PC based station because it is impossible to connect the Lidar directly to PC based station as the Lidar communicates via I2C. So, thanks to this STM32f407 board, we're able to retrieved the data from Lidar and then send it to PC based station via serial connection UART. On top of that, we're using STM32f407 for motor control by using PWM timer.

Here's the connection that we require during this project:

For UART	**For Motor**	**For I2C**
PA2 (TX)	PB8 (DIR)	PB6 (SCL)
PA3 (RX)	PB9 (STEP)	PB7 (SDA)

Lidar Lite v3 HP

This is High-performance Optical Distant Measurement Sensor which is Compact, lightweight, power-efficient ranging and proximity sensor with sturdy IPX7-rated housing for drone, robot or unmanned vehicle applications. It's also Easy-to-use 40 meter laser-based sensor offers greater than 1 kHz measurement speed up to 10 meter distances — and improved accuracy at all distances. User configurability allows adjustment between accuracy, operating range and measurement time. It communicates via I2C or PW, and also requires low power consumption; requires less than 85 milliamps during acquisition.

Exynos Odroid XU4

ODROID-XU4 is a new generation of computing device with more powerful, more energy-efficient hardware and a smaller form factor. Offering open source support, the board can run various flavors of Linux, including the latest Ubuntu 16.04 and Android 4.4 KitKat, 5.0 Lollipop and 7.1 Nougat.
By implementing the eMMC 5.0, USB 3.0 and Gigabit Ethernet interfaces, the ODROID-XU4 boasts amazing data transfer speeds, a feature that is increasingly required to support advanced processing power on ARM devices.

Driver A4988 and NEMA17

A4988 is a stepper motor driver that lets you control one bipolar stepper motor at up to 2 A output current per coil. Here are some of the driver's key features:

- Simple step and direction control interface
- Five different step resolutions: full-step, half-step, quarter-step, eighth-step, and sixteenth-step
- Short-to-ground and shorted-load protection

A NEMA 17 stepper motor is a stepper motor with a 1.7 x 1.7 inch (43.18 x 43.18 mm) faceplate. It is a bipolar that can work with 12V to have a 20 Ncm of torque. The step of this motor is 1.8 °.

c. Software

Keil µVision 5

The µVision IDE combines project management, run-time environment, build facilities, source code editing, and program debugging in a single powerful environment. µVision is easy-to-use and accelerates our embedded software development. So, we're using µVision in order to programme the STM32f407. Basically, for our project, we're going to need three main things which are a serial connection UART to send the Lidar data from STM32f407 to Exynos board, the i2c communication protocol which is be using to retrieve the data from the Lidar (STM32f407 as a master and Lidar as a slave), and last but not least, the generation of PWM (Pulse Width Modulation) for the NEMA17 stepper motor.

- **`USART Initialize()`:** initializes the UART with the transmission parameters (speed, flow control, send / receive mode, parity, stop bit, size of the transmitted data), then configure the transmit / receive pins (Tx / Rx) so as to redirect them to predefined pins GPIO, which will connect them to the RS232 connector.
- **`USART Puts(USART TypeDef* USARTx,volatile char *s)`:** transmits the message passed by the pointer *s.
- **`Moteur PWM Initialize()`:** initializes the GPIO pins for the generation of PWM for the stepper motor.
- **`I2C1 Initialize(void)`:** allows to initialize the device I2C1.
- **`I2C start(I2C TypeDef* I2Cx, uint8 t address, uint8 t direction)`:** allows to initialize a communication on the I2C1 device.
- **`get distance(void):`** This function consists of the configuration of Lidar and then get the data from Lidar. This is how it works :
  - send 0x00 (Device command)
  - send 0x01 (Hard reset and put all register at default value)
  - send 0x04 (Acquisition mode control)
  - send 0x08 (Default value for acquisition mode control)
  - send 0x02 (config max acquisition)
  - send 0x1d (max speed acquisition)
  - Pooling on bit 0 of the state register (0x01)
  - Read the data in 0x10 (Read first byte of distance)
  - Read the data in 0x0f (Read seconde byte of distance)
  - Concatenation of those two data to obtain the distance
- **`SysTick Handler(void):`** generates interrupt requests on a regular basis. Initialises and starts the System Tick Timer and its interrupt in the main. After this call, the SysTick timer creates interrupts with the specified time interval. Counter is in free running mode to generate periodical

interrupts. So first of all we set a variable *usTick* to 2500 because our stepper motor will move 2.5 ms for a step.

Last but not least, in the main function, we initialize all the components that we've been declared before including the System Tick Timer and its interrupt and then in the while loop, we will retrieved the value of distance captured by the Lidar and then transmits that value of distance captured by using the UART. Generally, for the best case, we need to retrieve the distance measured for each angle means that the Lidar and stepper motor need to be synchronized together but in our case the Lidar and stepper motor are working separately (Lidar in while loop and stepper motor in SysTick Handler) because the Lidar takes a little bit longer time to get a measure so it is difficult to be synchronized with stepper motor.

Termios Library

Termios is the newer Unix API for terminal I/O. The anatomy of a program performing serial I/O with the help of termios is as follows:

- Open serial device with standard Unix system call ***open()***
- Use standard Unix system calls ***read()*** and ***write()*** for reading from, and writing to the serial interface.
- Close device with the standard Unix system call ***close()*** when done.

We're using this library because we don't have software like Tera Term in Ubuntu in order to read the serial communication. By using this library, we'll be capable to retrieve the distance measured by the Lidar which is sent from STM32f407 by using UART. This is example of how it works:

This is the data that is being sent by the STM32f407 using UART

S	%d	\n
Header	distance measured	end of line

Basically, we want to retrieve the value of distance measured and thanks to the system call ***read()*** of the termios, we're capable to retrieve that value by taking out all the data between the header and end of line. First of all, the header will be detected first and once it's being detected, as long as it is not the end of line, all the data after the header will be stored in the array called ***int_buffer[]*** and then that array will be displayed on the terminal.

Robot Operating System (ROS)

For this project, we decide to use ROS as middleware. We can then use Google Cartographer directly from ROS. We just need to install ROS and follow by Google Cartographer.

In the Google Cartographer documentation site [1], we can find many demonstrations that can show how Google Cartographer works with ROS. After we install Google Cartographer, we run one of the demonstrations [2].

As the integration between the real time data and Google Cartographer is quite complicated, the generation of data is done separately and the data

retrieved will be shown in RVIZ, a 3D visualization tool available in ROS. This fake data will first be generated randomly without the use of serial data from STM32 microcontroller.

There are 2 major classes that are important for this part:

1. ros::NodeHandle
   - Write a node and interface to create publisher or subscriber in ROS
2. ros::Publisher
   - Manage an advertisement on a specific topic (here the topic is fakeLaserScan)
3. sensor_msgs::LaserScan
   - A subset of sensor_msgs message that interact with sensor. Contain some important variables for a laser scan like the minimum and maximum angle of the laser device rotation, interval between measurements, minimum and maximum distance of laser range device and etc.

All the documentation and the links related to this part can be found in appendix.

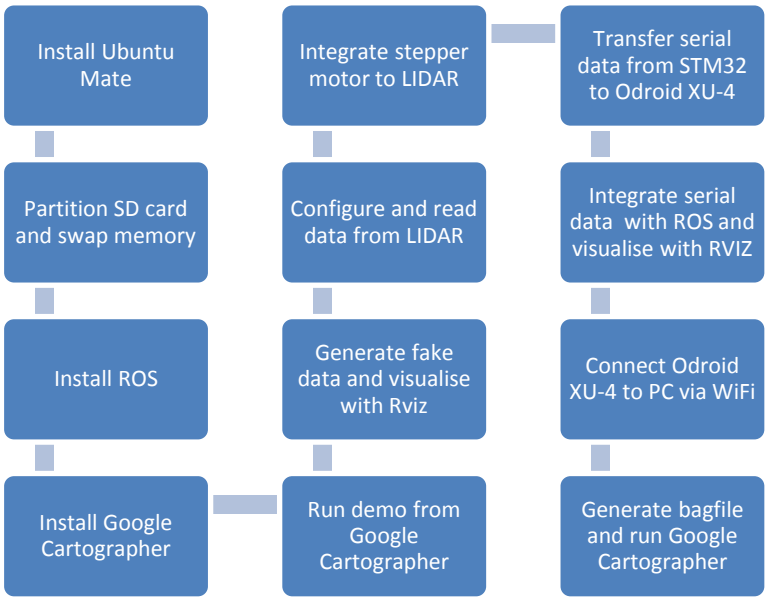
As mentioned before, we use Termios library to retrieve the serial data from STM32 microcontroller. The next step is to combine that part with ROS part through fakeLaserScan rostopic. Below you find some variables that needed to be determined according to the laser we have chosen.

The first step is to configure the serial data. Then, we need to declare some ROS objects like the rosnode and publisher for the rostopic. After that, in a loop where the rosnode is not going to exit, the program will start to read the serial data and convert the data from a string to a double. Then this data will be store in an intermediate array. Next, we need to populate the LaserScan message. Certain variables need to be modified according to the specification of our Lidar device and the NEMA motor that we use in this project. The message then will be passed through the publisher that we have declared before.

Finally, the program will be compiled and run in ROS and the command can be found in appendix.

III. TEST

There are several steps that need to be done in order to get a result that correspond to the objective.



Until the end of the project, we manage to finish integrate serial data with ROS and visualise this data with RVIZ. Some aspects like the performance of the LIDAR cause the project to be delayed. There are 2 steps that not yet be done: connect Odroid XU-4 to PC via WiFi and generate bagfile and run Google Cartographer.

IV. CONCLUSION

During this project, we are able to follow the objective of each task. However, due to several constraints like the performance of LIDAR, a lot of time has been spent to solve the problem. Currently, we are at the final stage of the project before we can make the indoor mapping. The important aspect of this project is done but somehow need a little bit of improvement in order for the system to work without any flaw.

V. APPENDIX

a. Install Ubuntu Mate for Exynos Odroid UX-4

i. Download the Ubuntu Mate ISO image from this link https://ubuntu-mate.org/download/ from your computer.
ii. Place this image in a SD card.
iii. Eject the SD card from your computer and put the SD card on the SD card slot of Exynos Odroid XU-4.
iv. Boot the board and follow the instruction.

For more information you can refer to this link:
https://linuxhint.com/install_ubuntu_mate_1804/

b. Partition for SD card and swap memory

The Exynos OdroidUX4 appears to be a powerful computing platform, but is there are any options for increasing the available RAM higher than 2GB because in order to run ROS we're going to need a quite powerful RAM.

It seems the ARM platform is SOC (system on chip). So the RAM is built-in the chip together with CPU and interface logic. Then, there's no possibility to add extra RAM but there is some program Linux uses to convert disk space into Virtual Memory by adding swap. So, in our case, we'll use the SD card in order to create a disk space that will be converted into virtual memory and then we'll add a swap. We're using 32 GB SD card so we've decided to divide it into 8GB for booting of Ubuntu and the rest will be used as virtual memory.

Here's is the link of the tutorial to create a partition on SD card:
https://askubuntu.com/questions/735096/how-to-format-a-partitioned-sdcard

Here's is the link of the tutorial of how to add swap:
https://www.digitalocean.com/community/tutorials/how-to-add-swap-space-on-ubuntu-16-04

c. Install ROS and Google Cartographer

ROS (Robot Operating System) provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. ROS is licensed under an open source, BSD license.

Cartographer is a system that provides real-time simultaneous localization and mapping (SLAM) in 2D and 3D across multiple platforms and sensor configurations.

The link how to install ROS Kinetic on Ubuntu:
http://wiki.ros.org/kinetic/Installation/Ubuntu

The link how to install and run the demo of Google cartographer:
https://google-cartographer-ros.readthedocs.io/en/latest/compilation.html

d. Run the project from ROS_Lidar_ws workspace and launch RVIZ

The link for the ROS package that publishes laser scan data:

Once you have modified the code in fakeLaserScan.cpp, you can follow the following command:

1- Compile the project (make sure the current repository is the repository of the workspace). In this project the the repository is at ROS_Lidar_ws/src/fakeLaserScan

```
$ catkin_make

$ source devel/setup.bash
```

2- Launch ROS

```
$ roslaunch beginner_tutorials fakeLaserScan.launch
```

You can find the package beginner_tutorials in Cmakelist in fakeLaserScan repository.

3- Check if the topic is publishing correctly

```
$ rostopic echo -n1 /fakeScan
```

4- Launch RVIZ

```
$ rosrun rviz rviz
```

For details use of RVIZ, you can refer to this link
http://www.theconstructsim.com/ros-qa-122-how-to-show-laser-data-on-rviz/