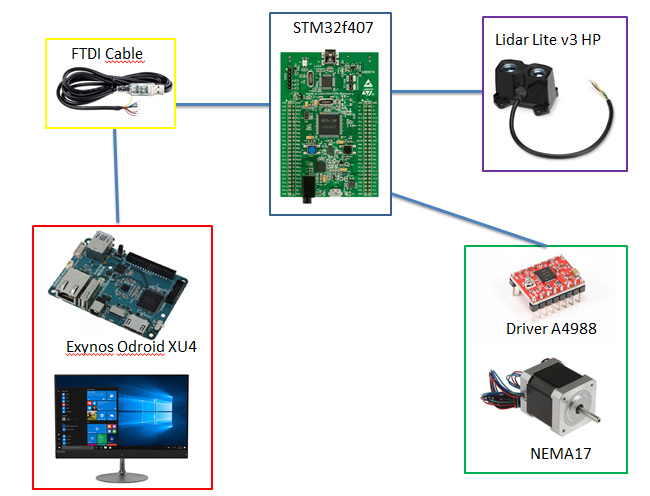
**Final Report Project GSE5 2018/19**

**BIN IBRAHIM** Ahmad Hanif Imran

**BIN SHAHIDAN** Muhammad Imran

1. **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:



PWM

UART

UART

I2C

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.

1. **CONCEPTION**
   1. **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.

* 1. **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

Driver A4988 and NEMA17

* 1. **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
  + send 0x01
  + send 0x04
  + send 0x08
  + 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 the data in 0x0f
  + 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 a 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)

1. **TEST AND RESULT**
2. **CONCLUSION**