Intrusion Detector for Domestic Applications

Embedded Systems

Warith HUSSAIN, Albane MALLET, Riheib NEMRI

Intro

Within the framework of the embedded systems course, we had to create a prototype of an Intrusion Detector for Domestic Applications. During the entire semester, our group tried to follow at our best the different instructions given to realise the best system possible. In this report, you will find explanations about the functioning of our system, the creation of the software and hardware part of our project as well as the course of the tests that we have carried out.

Table of contents

System's architecture

Goals and expectations

At the beginning of the project, a list of objectives have been given to us.
The main part was to develop a system capable of measuring a distance between our sensor and a target and the temperature of the room. The device created needed to be able to issue an alarm when a given limit was reached by one of the two sensors. Unfortunately, a lack of time and equipment didn't allow us to create the alarm.

The system is also connected to a keypad and a console that allows the user to program and operate the system. The LCD Screen permits to display the date, the time and the measures made by the sensors.

The keys of the keypad are programmed to perform certain tasks. As said previously, some of them are used to program the system while the others are operating the system.
To program the system, four keys are necessary, one for the setup of the date, one for the setup of the time, one to set the distance limit and another the temperature limit.
To operate the system, only three keys are required: one to start the operation, one to show the current time and one to show the information of the last alarm.

The last thing to know about this project is that it would be needed and requested to use FreeRTOS which is a real-time operating system often used for embedded systems.

Development of the hardware

The project was built around the launchpad Tiva TM4C123GXL (Annexe 1.1) which is a Texas Instrument board with a 32 bits 80MHz ARM Cortex processor. All the other devices will be connected through it. All of them use different protocols of communication. To determine the pins while respecting the communication protocols used, we relied on Annexe 2.

First of all, the two sensors are connected via the protocol of communication the Inter-Integrated Circuit (I^2C). To avoid too much issue with the wires and space we decided to connect the two sensors at the same boards' pins. The Tiva will be able to talk separately to each one of them via the I^2C because of their addresses which are different for one another.

The temperature sensor is a TMP101 (Annexe 1.2), a digital sensor with an accuracy of 1°C or 2°C and a temperature range of -55°C to 125°C.
The distance sensor is a VL53L0X (Annexe 1.3) which is a Lidar sensor with a good accuracy and a range up to 2 meters. Lidar is a method for measuring distances consisting in illuminating the target with laser light and measuring the reflection with a sensor.

The Keypad (Annexe 1.4) has 16 keys with 4 rows and 4 columns. We connected each of the rows and columns to different pins of the Tiva. The rows are connected as output pins and The columns are connected as inputs. They are also connected to a pull-down resistor which will return a value of 0 when no keys are pressed, the resistors are also meant to avoid reading issues or short circuits.

The LCD (Annexe 1.5) is a 16x2 dot matrix that can display alphanumerics, Japanese kana characters, and symbols. The device has 13 pins including 8 for the data. Our system uses only four of the data pins, the four higher. The other pins used permit the Tiva to power the display (Vcc /GND), enable it (EN), choose the register to transmit the data to (RS) and choose if we read or write on the LCD (R/W). Here, we only want to write on the interface. So, R/W should be equal to 0 and for that it's been connected to the ground.

All our hardware connections have been planned and resumed in a wiring schematics which can be found on Annexe 3.

SW architecture

As you saw in the last part, our project has many different elements connected with each other. To facilitate our understanding and the progress of our program with this multiplicity of devices, we have decided to split our code into different functions and categories.
Even though the different parts of the project are connected between them, we decided to make a category for each hardware piece of our system.

As requested, we tried to use FreeRTOS within our program. It will have served us to run at the same time our main program with all the hardware and our timer. As you can see in our task diagram (Annexe 4), we have created two tasks, one for the timer and one for the rest of the program. The two tasks are linked by a message sending the time from the timer (task 1) to the main program (task 2) when needed.

The Keypad is the main part of our project because he is the link between the user and our system and because it is through him that the actions are given. Each of its keys triggers an action programmed.
The LCD is the second most important part because it permits us to display the things needed for every action (the time, the temperature, the distance, etc...). Our LCD prints every character one by one. This is why we created two functions print. The first one is sending one character to the LCD for it to display and second one is sending the characters of an array to the first function.

Every function and action of our system start with the Keypad and finish printed on the LCD.

One of the actions programmed is to show the temperature of the room. It uses the temperature sensor via the protocol of communication I^2C. The sensor transmits an integer result. To display it on the LCD, we have found a code converting an integer into an array of characters.

The two others are similar. They allow the user to enter the date and time that he wants. The timer (task 1) is linked here because it will increment the time entered by the user every second to create a clock.

Test (evaluation of the system)

The first thing for us to do in this project was to understand and learn how to use the IDE Code Composer Studio and how to code in the language C. Fortunately for us, we already dealt with other IDEs and already code in other languages.
Being used to working with consoles, we tried to do the same with Code Composer. But, we didn't manage to open one in Code Composer to print any values. And we didn't know how to deal with code breakpoints.

So first of all, we started to program the LCD, because without it we couldn't verify if the other components work or not.
The Tiva has its own libraries so we needed to learn how to enable the pins and configure them.
In the Datasheet, there is a part to initialize the LCD so we just needed to follow it. At some point we needed to retry with different delays because it didn't react to certain commands.

For the Keypad, we just tried to make it react when any key was pressed, it worked for the two first columns but not with the 3rd and 4th one, so we needed to change the pins and connect them to an analog pin then it worked.
Then we needed to separate each key, which was easy when we understood how it worked.
The columns are in input mode so they receive a signal and the rows in output so they send a signal. We needed to verify which column received a signal then verify from which one it is receiving.
The columns always receive 0, when a key is pressed the corresponding column reacts. So now that we know which column it is, to know which line it is, we deactivate one by one the lines and verify if the column pin reacts, then we

know which line it is. Knowing the Row and the column, we know which key it is.

For the temperature Sensor, we needed to know how the I^2C of the Tiva worked, then to know the address of the TMP101. To understand the TIVA command we used the datasheet and the help of the internet. Then we found the register and the address for the TMP101 in the datasheet of the sensor.
For the Lidar, we didn't manage to code it but we tried to send a Register to various Addresses, then to receive something from various registers, but lack of time and knowledge, it didn't work.
With I^2C, when you get to a wrong address, it sends you an error number 255.

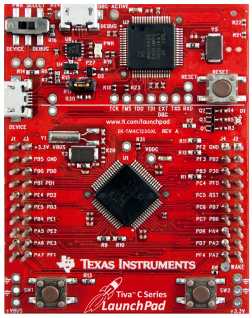
Finally , we tried to introduce FreeRTOS to our program. For that we took the example given for the TIVA, and modified it. We've put what we had in a task, and a time counter in another, to let the seconds work even if we do another thing. But it didn't work, all the delays used for the LCD made the seconds delay.

Conclusions

This project was for us a discovery of embedded systems, FreeRTOS, programming in C and the IDE Code Composer. But, we managed to get a project which is working. We encountered a lot of difficulties due in part to our lack of knowledge. That's why time has been a drag on us and we didn't have the time to meet all the expectations.
In spite of this, we are proud of the work we have done.

You will be able to find all our schematics and our program on our website (https://sites.google.com/view/project-alriwa/accueil?authuser=1)

Annexe

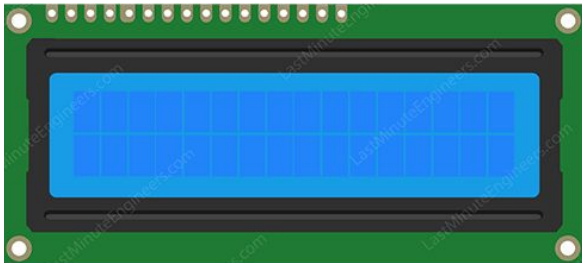


Annexe 1.1

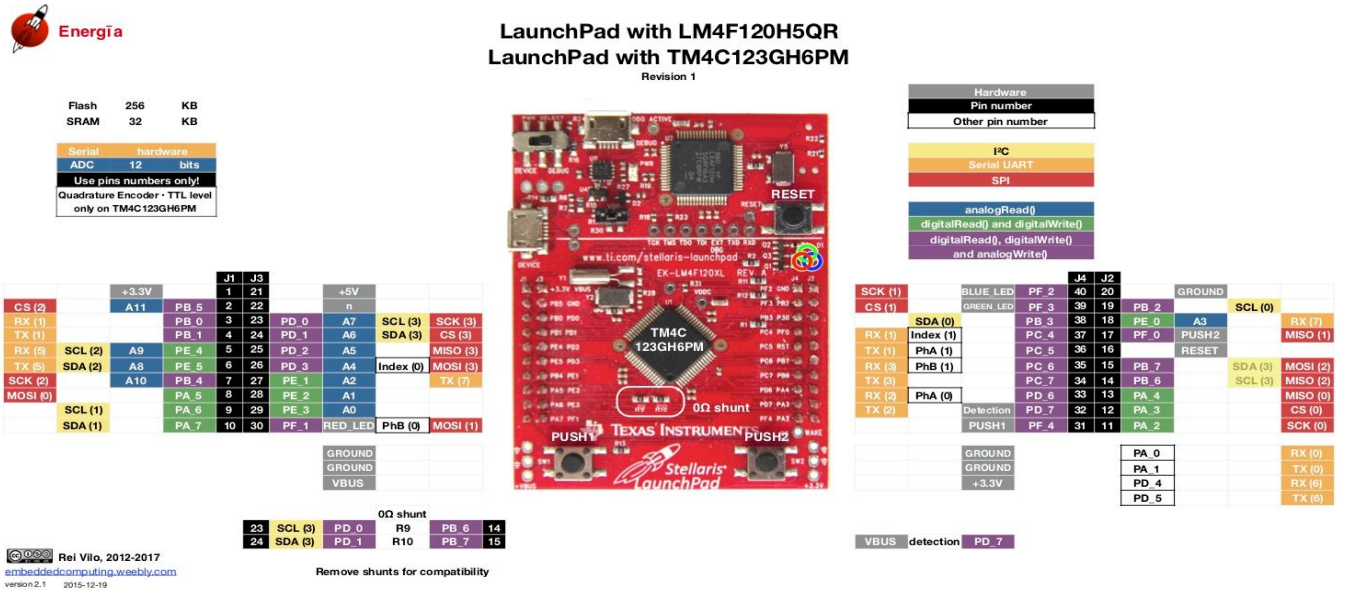
Annexe 1.2

Annexe 1.3

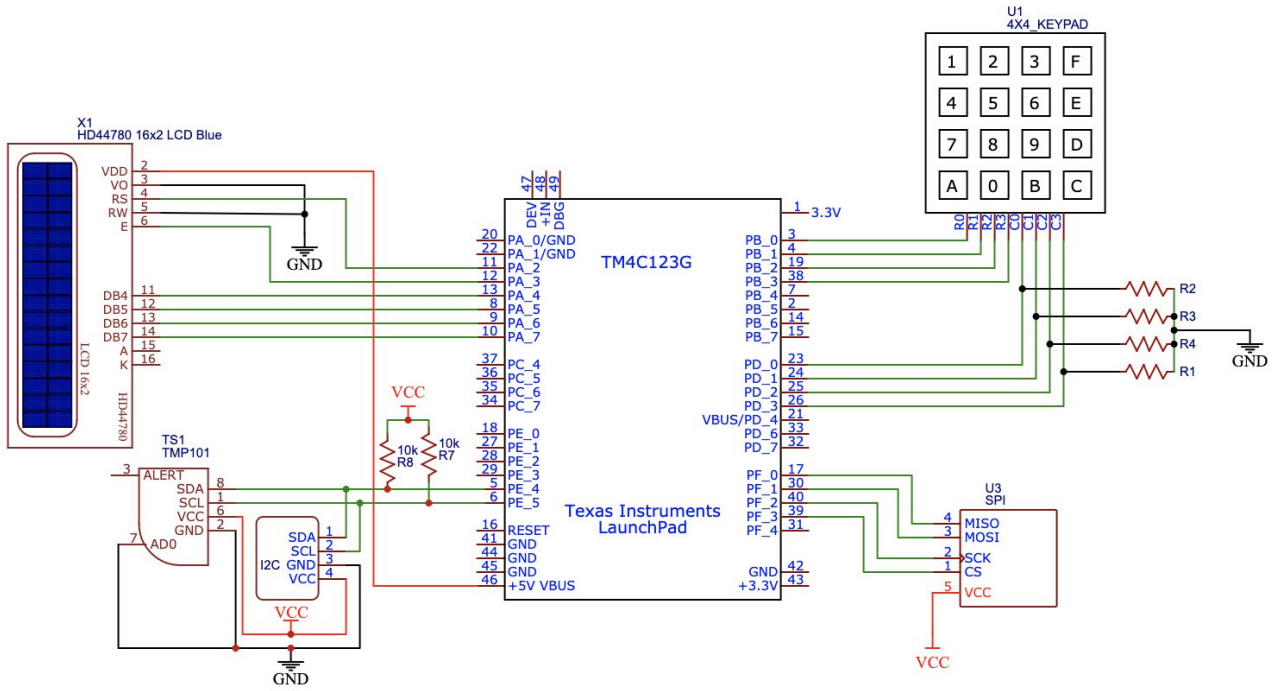
Annexe 1.4



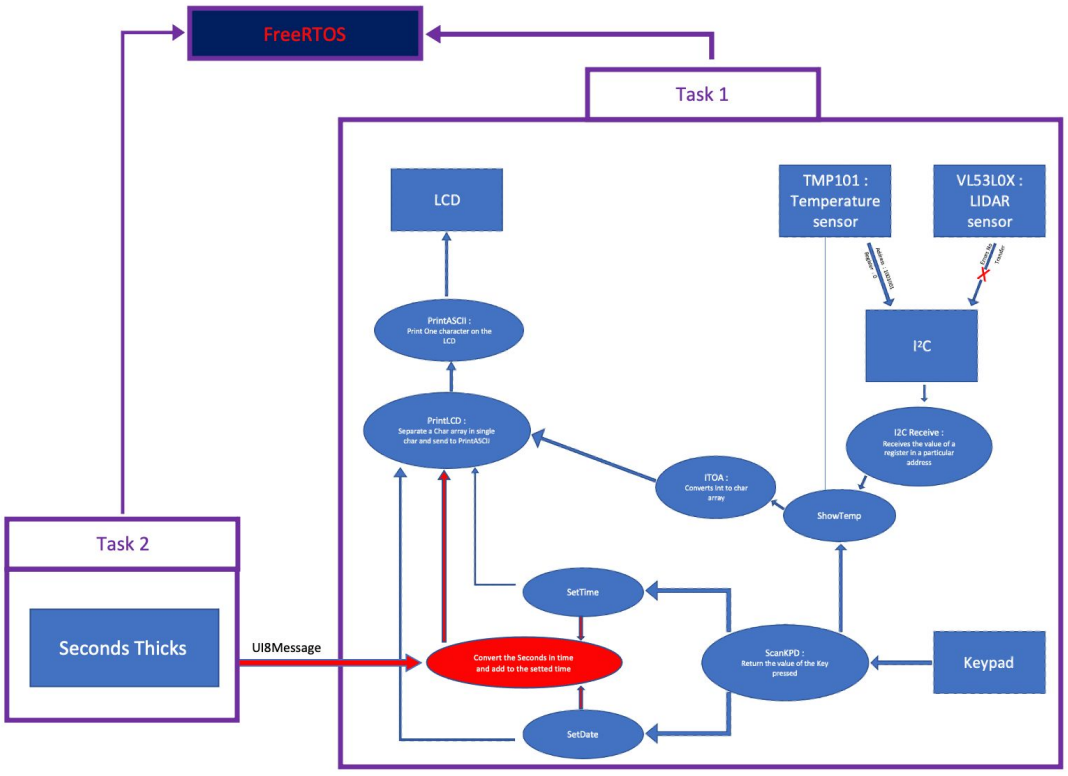
Annexe 1.5



Annexe 2



Annexe 3



Annexe 4