Course - Introduction to Embedded Systems

Exercise work – Alarm System

Video Demo: Demo_Video_Group_29.mp4

Thomas Kaatranen
Thomas.Kaatranen@student.lut.fi

1. Introduction

This project report will introduce an alarm system that has been built with the use of an Arduino UNO R3 and an Arduino MEGA 2560 R3 boards. The alarm system functions in a way in which, if the alarm is in its active state it will wait for movement to happen. When the system detects movement, it will start blaring the alarm and in order to disarm the alarm the user needs to input the correct four-digit password. The Arduino UNO and MEGA boards have been connected with a SPI communication protocol in which the UNO board acts as a slave and the MEGA board acts as the master. The basic communication happening between the slave and the master is the movement detection from the sensor. The system consists of different components such as a PIR sensor for the motion detection and an LCD that is used to communicate with the user through a visual interface. Additionally, the system has a buzzer and a keypad, the keypad is utilized for getting the user inputs.

1.1 State machine structure

This section will introduce the state machine structure of the introduced alarm system which can be seen in figure 1. In theory the alarm system has 5 different states however the disarm state is instantaneous after the correct password is given which will lead to the offline state. The different states of the alarm system are:

- Offline state
- Active state
- Alarm state
- Change password state

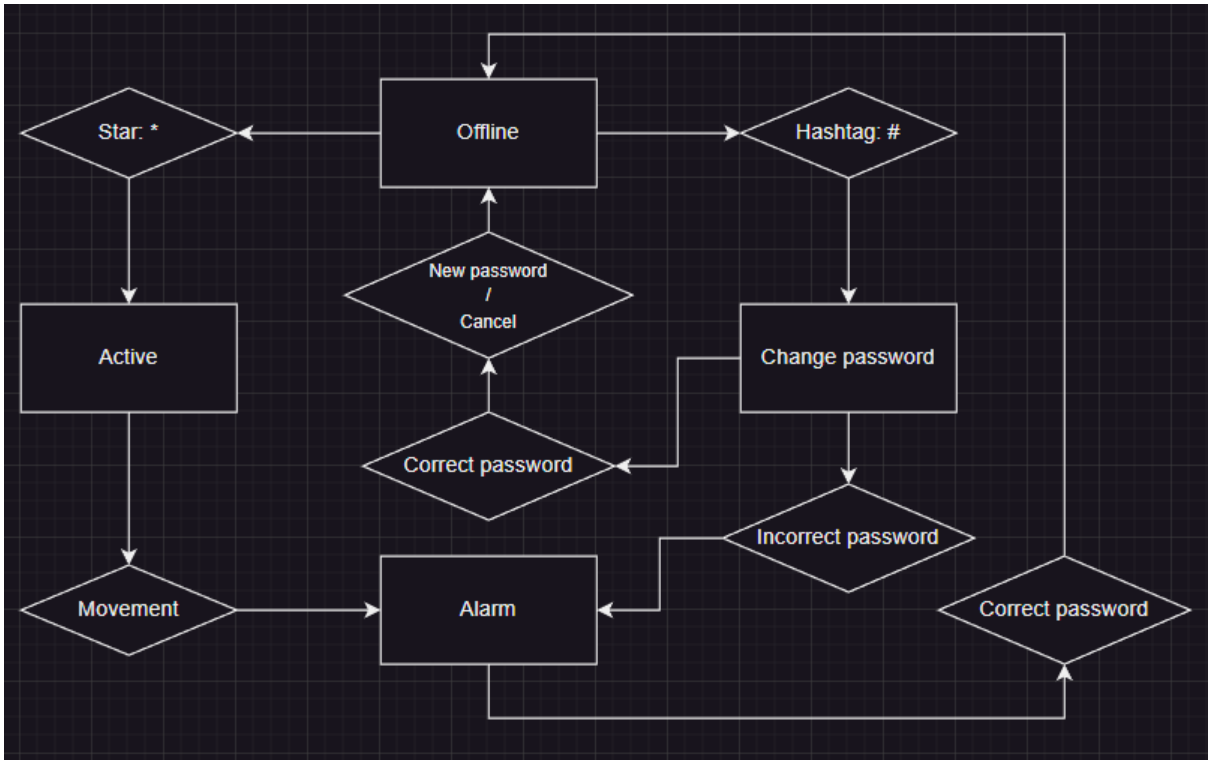

Figure 1. Diagram of the alarm's systems state machine structure

The state of the alarm system starts of as offline. From the offline state the system can either go into its active state or change password state, which is dependent on what input the user gives. If the user gives a star as an input the alarm system will go into its active mode. In the active mode the system will start monitoring for any movement with the use of a PIR sensor. When the sensor detects movement, the system will switch from the active state to its alarm state. When the alarm state is activated, the system will start blaring a buzzer until a correct password is given. When in alarm state and an incorrect password is given the system will inform the user through an LCD that the password is incorrect, and it will wait for a new input. Additionally, in the alarm state there are also interrupts happening. If the password is not given in a specific timeframe the system will reset any inputs that were given. When the correct password is given the buzzer will be disarmed and the alarm will go back to its offline state.

If a user gives hashtag (#) as an input, the system will navigate the user to the change password state. In the change password state, the user will first have to give the current password in order to move forward, however if an incorrect password is given the system will go into its alarm state. When the current password is given in the change password state the user will have the option to either cancel by pressing "B" and going back to the offline state

or the user can change the current password to a new one by pressing “A”. If the user gives “A” as an input the system will wait for a new password to be given. However, in this state the new password has to be a four-digit number. When a suitable password has been given the current password will change to the new one and the system will go back to its offline state where it can be activated.

2. System’s structure

This section will introduce the circuit schematic of the alarm system that can be seen in figure 2.

Circuit schematic

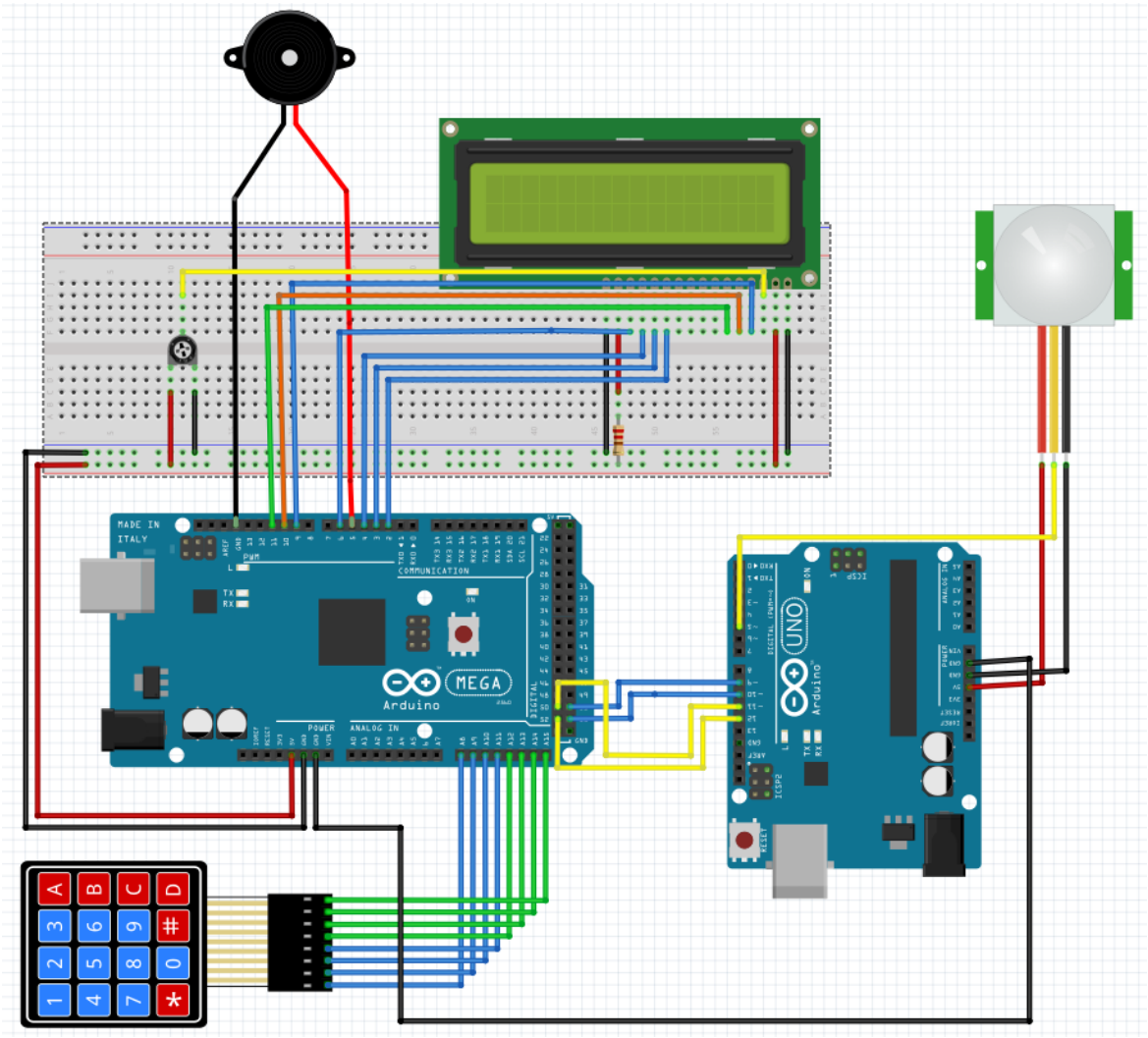

Figure 2. Circuit schematic of the alarm system

The components that were used in the circuit above are:

- Arduino UNO
- Arduino MEGA 2560
- LCD display
- PIR sensor
- 4x4 keypad
- Passive buzzer
- 10Ω Potentiometer
- 220Ω Resistor
- 34 Wires

3. The structure of the code

Most of the important functionalities will be found in the Mega board's code. The Mega board's code consists of a main loop that loops the whole program. The loop itself has four different states like it was already mentioned state machine structure section. The first state in the code is "case 0" which is the offline state. The offline state displays information to the user through the lcd display and waits for a key input. When the user gives an input of a star (*) it will navigate the code to "case 1" which is the alarm active state. When a user gives hashtag (#) as an input the code will move to "case 2" which is change password state.

In "case 1" the lcd will display * ALARM ON * to the user and wait for movement to be detected. The movement detection functionality is located in the UNO board's code which has a while loop that either sends an empty message meaning there has been no movement or it sends a message of "movement_detected" which indicates that the pir sensor has detected movement. In this case the UNO board sends that message to the MEGA board with the use of the SPI communication protocol, when a message is received the MEGA board's code will compare the received message to the "movement_detected" message. If the messages are the same the code will jump to "case 3" which is the alarm active state. If the message does not equal "movement_detected" the loop continues. In the alarm active state, the code will first display the following information for the user through the lcd "Alarm is active" and "Press # - Disarm" and then it goes to an infinite while loop. In the while loop the code waits for an input to happen, if the hashtag (#) keypad is pressed the code will call "inputPassword()" function and pass the current password to it as a parameter.

The input password is a function that displays “Enter Password:” for the user and the user can give number-based keypad inputs to fill the password. As the keypad inputs are given the lcd will display the given inputs, the inputs can also be removed by pressing the keypad input C which in the code is the if statement “if (key == 67)”. In order for the user to give a password the user will have to give D as a keypad input which is the if statement “else if (key == 68)”. In this if statement there are three checks that check whether the given password is correct in terms of its length. The only viable option is to give a password that is the length of four digits. When a password that is four digits is given the “inputPassword()” function will set outcome as 1 and break out and then return the outcome to case 3 in which the function was called earlier, the outcome can either be 1 which means successful or 0 which means unsuccessful. If the returned outcome from the “inputPassword()” function equals 0 the “inputPassword()” function is called again, only when the outcome is returned and equals 1 the code will set state to 0 and break out of the “case 3” which will traverse the alarm system back to “case 0” which is the offline state.

When a user gives hashtag (#) as a keypad input in “case 0” the code will set state to equal 2 which will cause the code to jump to “case 2”. The “case 2” (change password state) will display messages “Redirecting to” and “Change Password” with 2 second delay and then call a function called “inputPassword()” and send the current password as a parameter to that function. When the function is called if the user returned outcome equals 0 the code will navigate to “case 3” and go through that process again and if the returned outcome equals 1 the code will move on in the current case which is “case 2”. In the case the user can either back out by giving a keypad input B which will traverse the code back to “case 0” or the user can give a keypad input A which will call a changePassword() function that will be given the current password as a parameter. The “changePassword()” function works the same way as the “inputPassword()” function in which it checks whether the given keypad inputs are numbers and is the given passwords length four digits. If the given password does not equal to four digits by length the lcd will display an instructing message to guide the user to give a correct password. If the given password equals to four digits by length the code will break out of the functions loop and return the given password as a return value. The return value is then copied to the password variable which overrides the old password with the new one. When the copying is complete the code will set state to equal 0 which will navigate the code back to “case 0”.

4. Conclusions

The introduced alarm system follows the basic structure of any other alarm. However, it does not strictly follow the guidelines that were given in the exercise work explanation since there was given a freedom for modifying the system in a way that it still behaved like any other alarm system would. The code is structured into the MEGA and UNO boards files which communicate with one another in a certain state. Additionally, the code follows the use of Embedded C coding standard, and it is comprehensively commented in order to help any code reviewers to understand what is happening in each part. Overall, the alarm system fulfils all the minimum requirements that were set for the exercise work. In addition to the minimum requirements there are some additional functionalities that were implemented, such as:

- Information is displayed to user via LCD.
- Use of interrupts.
- Possibility to arm/ream the system.
- Adding your own justified extra feature – Change password.
- User-friendly UI.
- Use of good coding practices (Embedded C coding standard can be used as a reference).

Video Demo: Demo_Video_Group_29.mp4