Open your mind. LUT.
Lappeenranta University of Technology

BL40A1812 Introduction to Embedded Systems

Final Project

Introduction

This is the report for the final project of LUT University course “BL40A1812 Introduction to Embedded Systems”. This report aims to describe the circuit diagram of the final product, and explain what our team did and why, and what could have been improved.

1. EXPLANATION OF THE SYSTEM

The system is a simplified model of an alarm system with a timer, motion sensor, buzzer, and a keypad to input a password with. Upon starting the program, the system waits in idle state until any movement is detected by the motion sensor. Upon triggering the sensor, the system starts the 10 second timer (not shown to the user) and waits for the user to input the correct password using the keypad. If the user enters the correct password in the 10 second timeframe, the alarm is disarmed. The user is then prompted to either rearm the system, starting the program anew, or exiting the program altogether.

If the user either enters the wrong password or the timer counts to 10, the alarm is activated, turning the buzzer on. The user is then once again instructed to enter the correct password, but this time with no timer nor a penalty for inputting the wrong password. Upon entering the correct password successfully, the alarm is disarmed, turning the buzzer off and prompting the user to either rearm the system or exit.

The system also features an LCD screen to give the user instructions on what to do and notifications on what the system is currently doing.

2. Circuit and state machine diagrams

Below is the circuit diagram and state machine diagrams for the system. The ATmega2560 (“Mega”) functions as the master and is responsible for most of the code and functionality for the entire system. The ATmega328p (“Uno”) is a slave, receiving instructions from the master (via SPI) on when to turn on and off the buzzer.

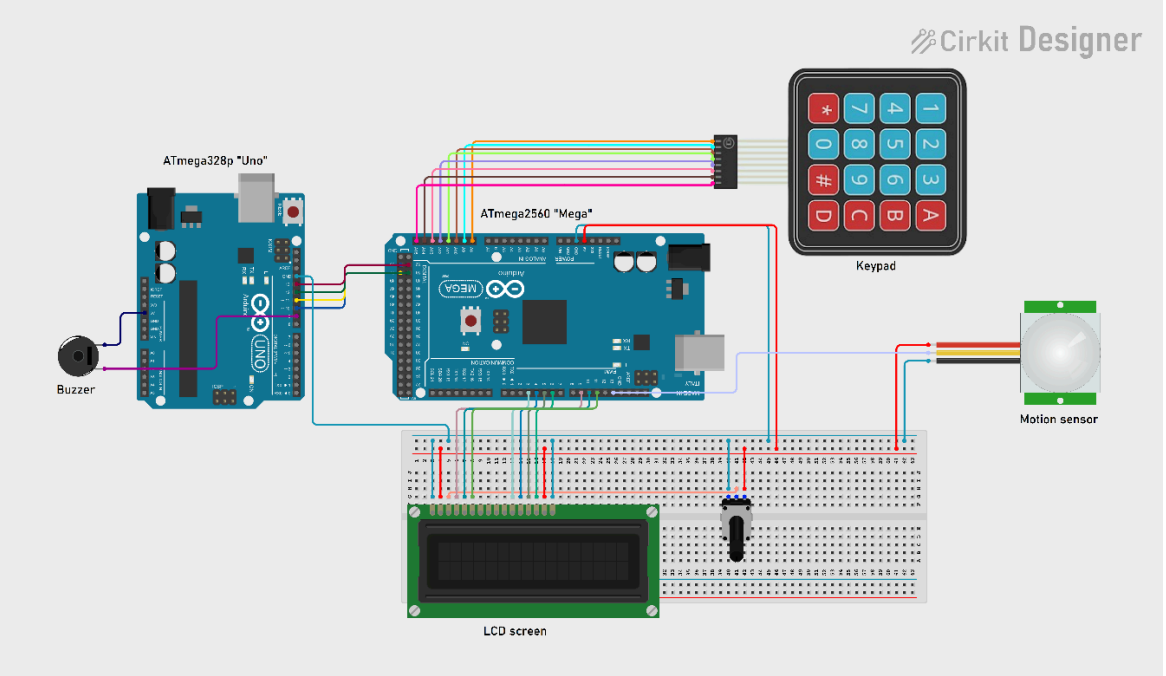

Figure 1: Circuit diagram of the final project

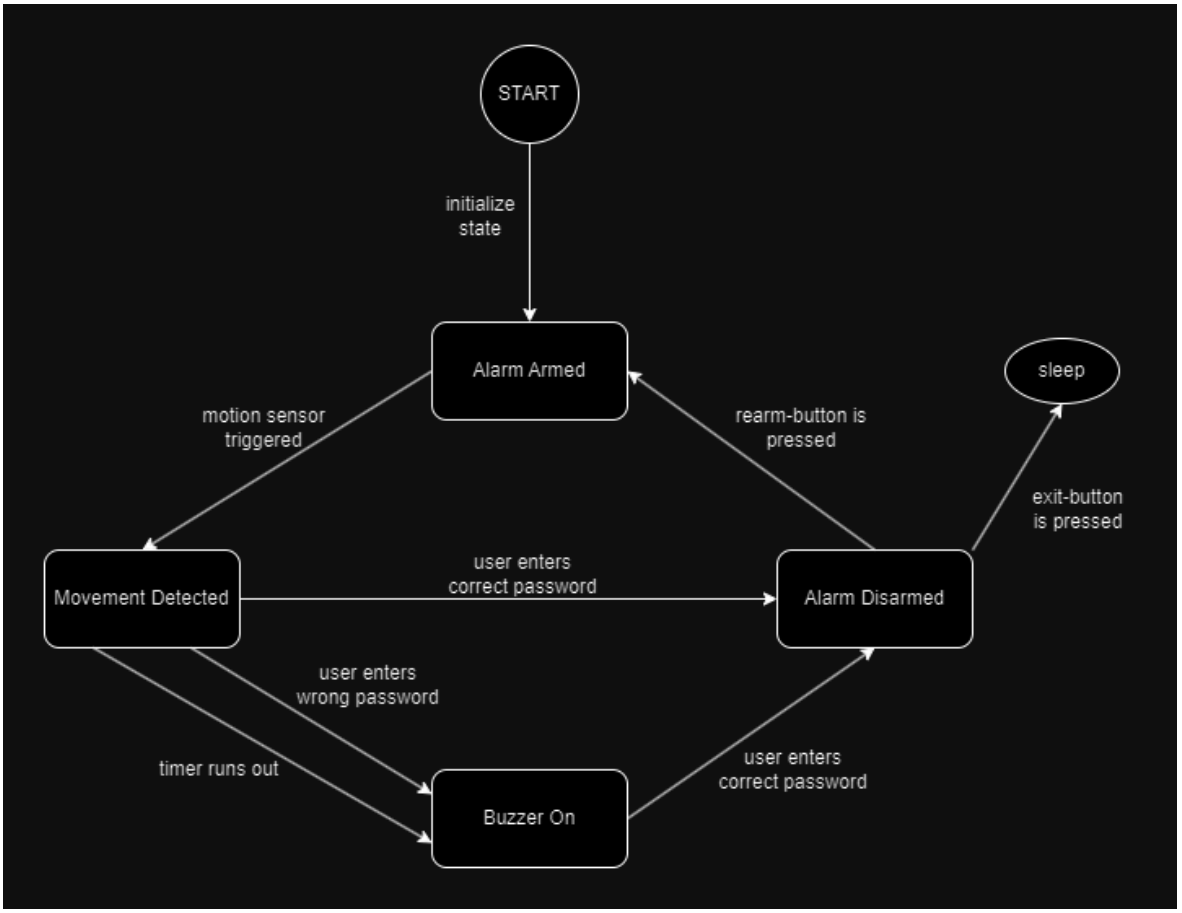

Figure 2: State machine diagram for the master (ATmega2560)

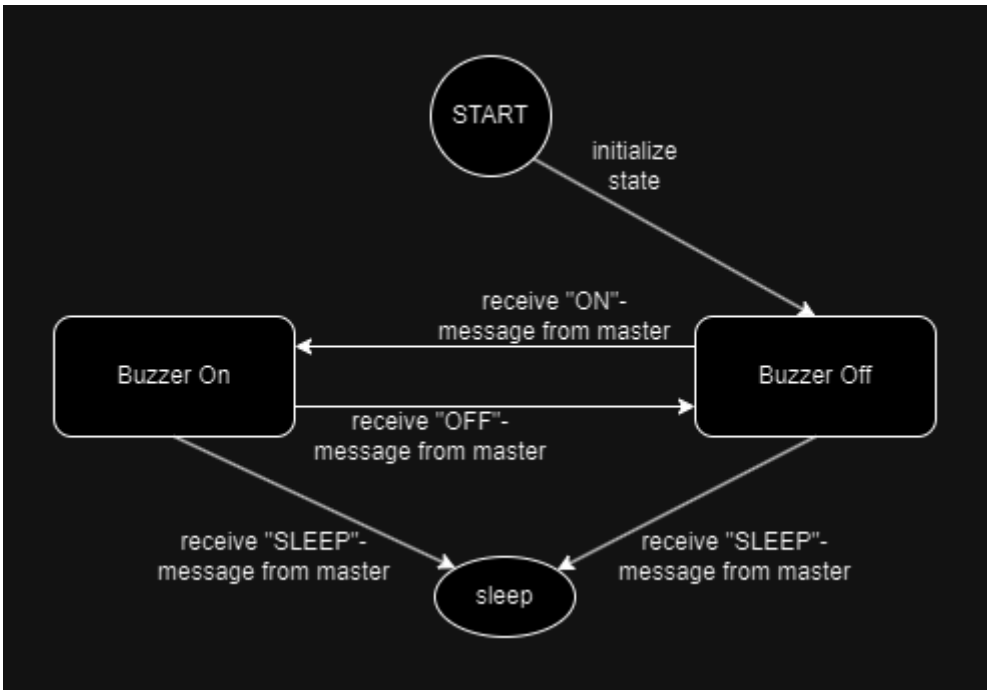

Figure 3: State machine diagram for the slave (ATmega328p)

2.1 Circuit diagram

The connections and electronics were taken from the course exercises' tasks, their instructions, and example answers. The connections for the keypad, SPI communication and the LCD screen were taken from course examples. LCD connections from week 8, SPI communication from week 10 and keypad from the additional materials section. The buzzer is connected to the ground and pin 9 of UNO while the motion sensor takes power from the MEGA board and communicates via pin 13. All devices are connected to a common ground. The two controllers themselves are connected to each other and communicate using SPI.

2.2 State machine diagrams

Figure 2. shows the state machine diagram for the master, "Mega". The first state "Alarm Armed" acts as an idle state of sorts, doing nothing until the motion sensor gets triggered. Following this, the state machine moves quite linearly through the rest of the states, until the entire system is either rearmed, resetting to idle state in "Alarm Armed", or entering sleep mode. The sleep node in the diagram has no outgoing arrows as the functionality was added last-minute and we didn't implement a wake-up routine. Ideally the sleeping state would be joined by an arrow to the "Alarm Armed"-state, or there would be a state where the program is shut down entirely.

Figure 3. shows the state machine diagram for the slave, "Uno". Since the master has most of the functionalities of the entire system, Uno is only responsible for turning on and off the buzzer. It is constantly listening for the master's messages and reacts accordingly. After being initialized in the "Buzzer Off" -state, the buzzer is either turned on or off depending on the user's correct or incorrect actions. If at any point it receives the "SLEEP"-message, the entire system is put into the sleep state, but with no way out as mentioned previously.

In reality, the system can only enter this sleeping state after the master has been through the “Alarm Disarmed”-state, since this is the only place in the code where the master can send the sleep-message.

3. REQUIREMENTS

Since our team had little-to-none prior experience working with breadboards and microprocessors etc. we decided to get the minimal requirements working first and then adding extra functionality if we had the time. Below is a table containing all the features implemented, and the proposed point values as shown on the grading file.

Since the sleep mode-functionality was only partially implemented, the TA deemed it worth 2 points instead of the typical 3.

FUNCTIONALITY	POINTS
minimum requirements	10
information is displayed via LCD	2
use of interrupts	1
possibility to rearm the system	3
use of sleep modes	2
TOTAL	18

Table 1: Implemented features and their point values.

The minimum requirements contain everything listed on the exercise work instructions -file. These include:

- motion detection
- buzzer for the alarm (using PWM)
- SPI communication between MEGA and UNO
- 4 number password using the keypad
- “submit password”- and “backspace” -buttons
- timer before the alarm sounds (not displayed)
- the system verifies the password, and informs the user
- correct password disarms the alarm, and an incorrect password/timeout activates it
- use of state machine structure

The additional functionalities are something we thought would be simple enough to implement before the deadline. There were talks of adding more, but they turned out to be too complicated or take too much time.

Implemented features:

- The LCD screen is used to display everything relevant to the user, giving instructions, verifying the password and informing them on what the system is doing at the moment.

- Interrupts are used when implementing the 10 second timer system, triggering the alarm if the time runs out without a correct password.

- When the alarm is successfully disarmed (or the correct password is given before the alarm sounds), the user is prompted to either rearm the system, or exit altogether. Rearming the system resets the timer, motion sensor and all relevant registers, and begins anew, waiting for the motion sensor to detect something again.

- The sleep mode is used when the user wants to stop using the program in the same way the rearm functionality is presented. However, there's no way to leave the sleep state apart from resetting the microprocessors with the on-board reset-buttons.

4. IMPROVEMENTS

As previously mentioned, our group had minimal knowledge working with these kinds of electronics, so the entire project was a learning experience for all of us. We all attended most of the exercises as a group, mostly completing the easiest tasks of each one while getting familiar with our hardware. A lot of time during the exercises and our final project was spent on debugging, googling, and trial-and-error, all of which would have probably been lessened had we fiddled and practised with the boards in our free time.

We also started to work on the project quite late in the course, as we felt the deadline being at the start of May would give us well enough time to have the minimum requirements down. In reality, we met once a week for the start, and finished the project on 3 consecutives days during the last week, just a few days before the deadline. Had we started earlier, we could have gotten help from the TAs and possibly had more time to implement additional functionalities.

All in all, we learned quite a bit about microprocessors and electronics over the lifetime of this course, and got to work on more lower level programming (and using C), instead of the higher level languages most of us are accustomed to. With more experience in C in our own projects or free-time tinkering, the course work would have probably have been easier and more rewarding.