Open your mind. LUT.
Lappeenranta University of Technology

Embedded exercise work

Konsta Jalkanen 000489689
Teemu Hiltunen 000393597
Lauri Heiskanen 001135864
Group 03
Elegoo nro 14
Mega nro 13

Bachelor’s thesis
LUT School of Energy Systems
Electrical Engineering

5 May 2024

1. INTRODUCTION

This is the report for the exercise work of the embedded systems course where the exercise work is to make a motion detection system with two Arduinos. It includes a list of features and their descriptions, the overall architecture of the circuit and structure of the made code, what was good and bad, and improvements to our system.

2. FEATURES

Here’s a table 1.1 containing all features with a short description of it. The feature is marked “additional” if it is from the list of the additional features and “extra” if it’s a feature of our making:

Table 1.1 Implemented features with their description.

Name	description
Backspace button	Button D on the keypad for deleting the last character of the password
Buzzer alarm	Alarm when the password wasn’t correctly inputted or 10 seconds were passed
Communication check (additional)	The system checks that communication can be found
Communication information (extra)	This extra feature displays using LED users whether the connection is found and formed or not. These are connected to mega and explained in section 3.2.2
Disarm	Correct disarms the alarm and the countdown ends.
I2C/TWI connection	Connection between Mega (master) and Uno (Slave)
Information is displayed via LCD (additional)	States of the system and password inputs are displayed on the LCD
Interrupts (additional)	The system uses interrupts as timers on mega.
Keypad	For inputting passwords
Motion detection	State machine proceeds to motion motion-detected state
Password after timeout	The alarm ends when the correct password is inserted after a timeout
Password check	Causes an alarm for the wrong password. Correct disarms
Rearm (additional)	The system can be rearmed with a button.
Rearm only when the alarm is off (extra)	The only way to rearm is after inserting the correct password. See section 4.3.
State information (extra)	This is an extra feature that extends the countdown information feature. The system displays states armed and unarmed.
State machine	Simple state machine up help by mega is implemented with four states: Armed, Movement Detected, Alarm, and Disarmed
States of countdown informed to user	Information of whether the password is correct, incorrect, or the time has run out is displayed by LCD
Submit button	Button A on the keypad for password submission
Timer for the alarm (10 secs)	Count down for the alarm

They are discussed in more detail in the following sections.

3. ARCHITECTURE AND STRUCTURE

In this section, we discuss our structural and architectural choices. We aimed to simplify the system with the least number of moving parts as they tend to be most easy to debug, implement, and read.

3.1 Structure of the code

Our uno and mega codes are made of multiple files and are compiled into a single code with a make file. Both's main structure of code is "main.c" which contains function calls of other functions, code for the communication between devices (I2C/TWI), and deciding code that activates desired code blocks as needed. Other code is divided by the function of their files.

The code is well commented on and is up to the standards of *"Embedded C coding standard"* by Michael Barr as instructed. The code utilizes functions for all, and a singular function does only one function as recommended. The naming of variables is clear and follows the standards.

3.1.1 Arduino Uno

Arduino Uno is responsible for displaying information for the user and playing the alarm. Arduino Uno works as a slave in communication. It waits for a message from Mega, parses it, and acts accordingly.

Arduino Uno displays the current state of the system, sent by Mega, as text on the LCD screen and the state of the alarm plays sound via the buzzer. This is done according to the instructions for the exercise work.

3.1.2 Arduino mega

Arduino Mega is coded to be a state machine responsible for collecting input from a motion detector, keeping a timer for alarm, and checking user password input. It works as the master in the communication sending displayable information to the slave. The timer works as an interrupt when ten seconds have passed.

As information must be displayed, it sends messages to uno to display the information. The system displays the state changes or when a user writes correct or incorrect passwords.

3.2 Architecture of the circuit

Uno and Mega are connected with an I2C/TWI connection as we deemed it to be enough for our exercise work. The following sections explain precisely what and why something is connected to Arduino.
The following diagram is the circuit diagram for a quick show:

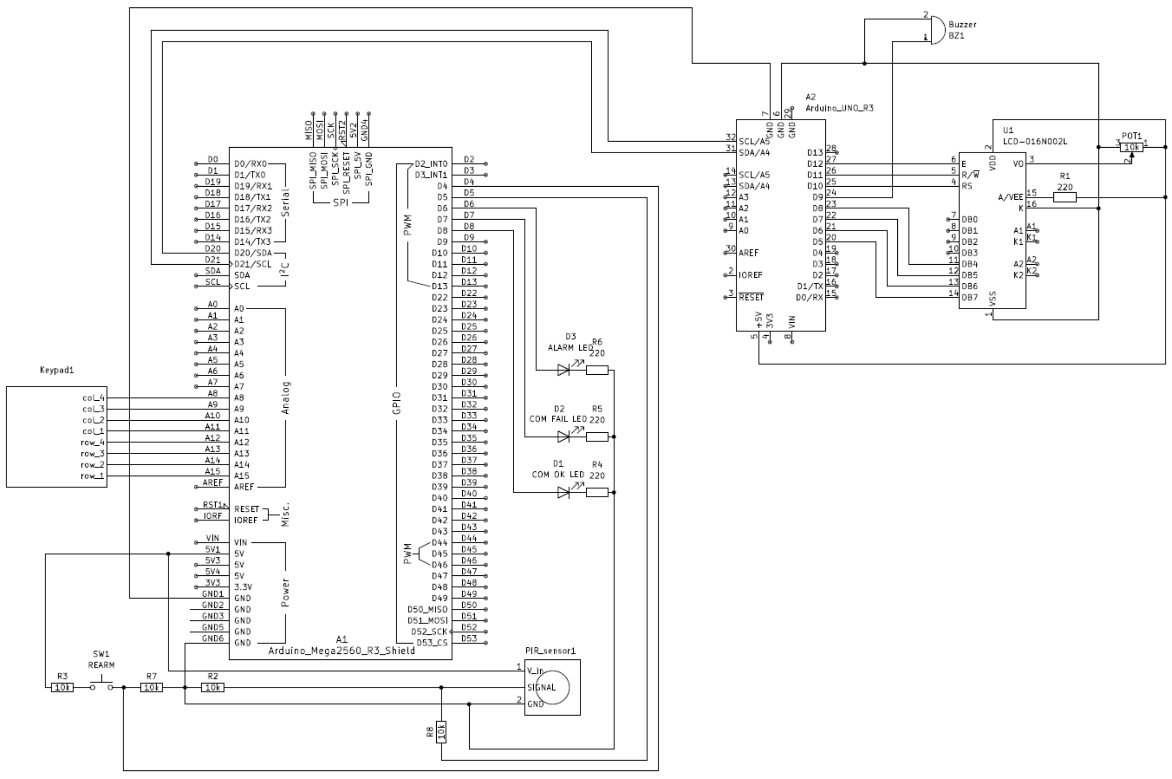

Figure 1.1 Circuit diagram

The circuit diagram is also summited in pdf form as instructed.

3.2.1 Arduino Uno

Arduino Uno is connected to Mega, LCD screen, and buzzer. This is because Arduino Uno was chosen to control the display for information to the user as displaying doesn't require as many pins as reading sensors.

As Uno receives a notification about a state change or password it displays it through an LCD screen. When the alarm state is activated uno starts playing the alarm sound via the buzzer. As Mega moves to the next state and Uno receives information about that, it stops the alarm.

3.2.2 Arduino mega

Arduino Mega is connected to Uno, three LEDs, a motion detector, Rearm, and a keypad. This is because Mega was chosen to be an input controller and user of user input. User input is received through a motion detector and keypad.

Arduino Mega has its own 3 LED lights. Two for indicating if communication is found and one for when the alarm is going off. These lights are used for debugging and indicate to the user the current state of communication as uno itself doesn't "test" the connection but only listens. The third LED is connected to Mega so it could display to the user that Mega is in an alarm state to know that Mega and Uno are cooperating.

In the demonstration video, viewers can observe that the use of the rearm button had some problems which probably were with the connection between the breadboard and wires or between the breadboard and legs of the button. This resulted in not always using the button for rearming and connecting a wire to the incoming resistor.

4. MISTAKES AND IMPROVEMENTS.

As this exercise work nears the end, we think that our overall work is good and follows the instructions almost to the point. There were a few mistakes, some unimplemented features, and one feature changed from the instructions.

4.1 Mistakes

There were a few mistakes not properly demonstrated in the demonstration video. The most notable one was the rearm button not always working. We guess that the breadboard is loose and thus not properly connected to wires and buttons. This was fixed by skipping the button with the wire. Another video mistake is the demonstration and explanation of the backspace button. In our system, pressing the backspace causes the iteration of written code. The last found character's position is set to be the next writable position so the next number will override the last number if pressed multiple times earlier position.

4.2 Unimplemented features

We had some more ideas about what could be implemented to the system but as the deadline grew closer, we decided that we wouldn't implement these. Some of them were useful but not necessarily needed and one was more of a joke. Here's a short list of the unimplemented features:

Table 1.2 Implemented features with their description.

Name	Description
Bad Apple	If the user inputs the correct password, Uno will play part of the song Bad Apple.
Displaying passwords	Uno would have displayed the letters of the password when typed in.
Larger passwords.	Passwords could have been between 4-8 characters.
LEDs for state display.	Uno would have led for each state and lit up the current one. Unnecessary as the LCD screen does the same.
Password change state	A state where the user could change the password even without EEPROM.

4.3 Changed feature.

We changed the rearm to only be able to be activated when the password is inputted correctly as it is an additional way to break the security, as the system would probably be used as a security device. Shutting down the alarm by rearming could be used as a loophole in a fault injection attack by an intruder as inputting current in a specific wire would disable the alarm.

4.4 Improvements

There are some obvious improvements to the system like adding the rest of the additional functionalities or the features mentioned in the unimplemented list as more features. Here are upgrades to the current system:

- Uno could have the buzzer be interrupt driven so playing changing sounds would be possible while operating others normally.
- Inputting the wrong password doesn’t cause the timer to stop which might lead to times up. This doesn’t cause problems but causes uno to display time-up text on LCD.
- The alarm is only one annoying note which could be replaced with another song.
- If a communication error occurs it does not restore it and Mega must be restarted

5. CONCLUSIONS

The overall work was made as instructed and came out well. The system mostly worked as intended and we didn’t have the same problems as with the weekly exercises. We made most of the additional functionalities and three extra self-justified functions, although small but handy. During the exercise, we learned to use I2C properly and design of own system out of embedded microcontrollers.

REFERENCES

Barr M. *Embedded C coding standard* [Online]. [Accessed 4.5.2024]. Available at https://barrgroup.com/embedded-systems/books/embedded-c-coding-standard

Github Repository:
https://github.com/Dorfultariant/arduino_embedded