Open your mind. LUT.
Lappeenranta University of Technology

Embedded exercise work

Konsta Jalkanen 000489689
Teemu Hiltunen 000393597
Lauri Heiskanen 001135864

Bachelor's thesis
LUT School of Energy Systems
Electrical Engineering

4 May 2024

1. INTRODUCTION

This is the report for the exercise work of embedded systems course where the exercise work is to make motion detection system with two Arduinos. It includes list of features and their descriptions, 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. Feature is marked "additional" if it is from additional features list 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 keypad for deleting last character of the password
Buzzer alarm	Alarm when password wasn't correctly inputted, or 10 seconds was passed
Communication check (additional)	System checks that communication can be found
Communication information (extra)	This extra feature displays to using LEDs user whether connection is found and formed or not found. These are connected to mega and explained in section 3.2.2
Disarm	Correct disarms the alarm and 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 in the lcd
Interrupts (additional)	System uses interrupts as timer on mega.
Keypad	For inputting passwords
Motion detection	State machine proceeds to motion detected state
Password after timeout	Alarm ends when correct password is inserted after timeout
Password check	Causes alarm for wrong password. Correct disarms
Rearm (additional)	System can be rearmed with button.
Rearm only when alarm is off (extra)	Only way to rearm is after inserting correct password. See section 4.3.
State information (extra)	This is an extra feature that extends the countdown information feature that the system constantly displays states armed and unarmed too.
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 password is correct, incorrect or time has run out is displayed by lcd
Submit button	Button A on keypad for password submission
Timer for the alarm (10 secs)	Count down for the alarm

They are discussed in more detail in following sections.

3. ARCHITECTURE AND STRUCTURE

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

3.1 Structure of the code

Both of our codes for uno and mega are made of multiple number of files with are compiled into 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 actives desired code blocks as needed. Other code is divided by the function to their own files.

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

3.1.1 Arduino uno

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

Arduino Uno displays current state of the system, send by mega, as a text in the lcd screen and in state of 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 motion detector, keeping timer for alarm, and checking user password input. It works as master in the communication sending displayable information to slave. Timer is interrupt driven but otherwise it acts as busy driven.

As information must be displayed, it sends messages to uno to display the information. This happens when state changes and when user submits both correct and incorrect password.

3.2 Architecture of the circuit

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

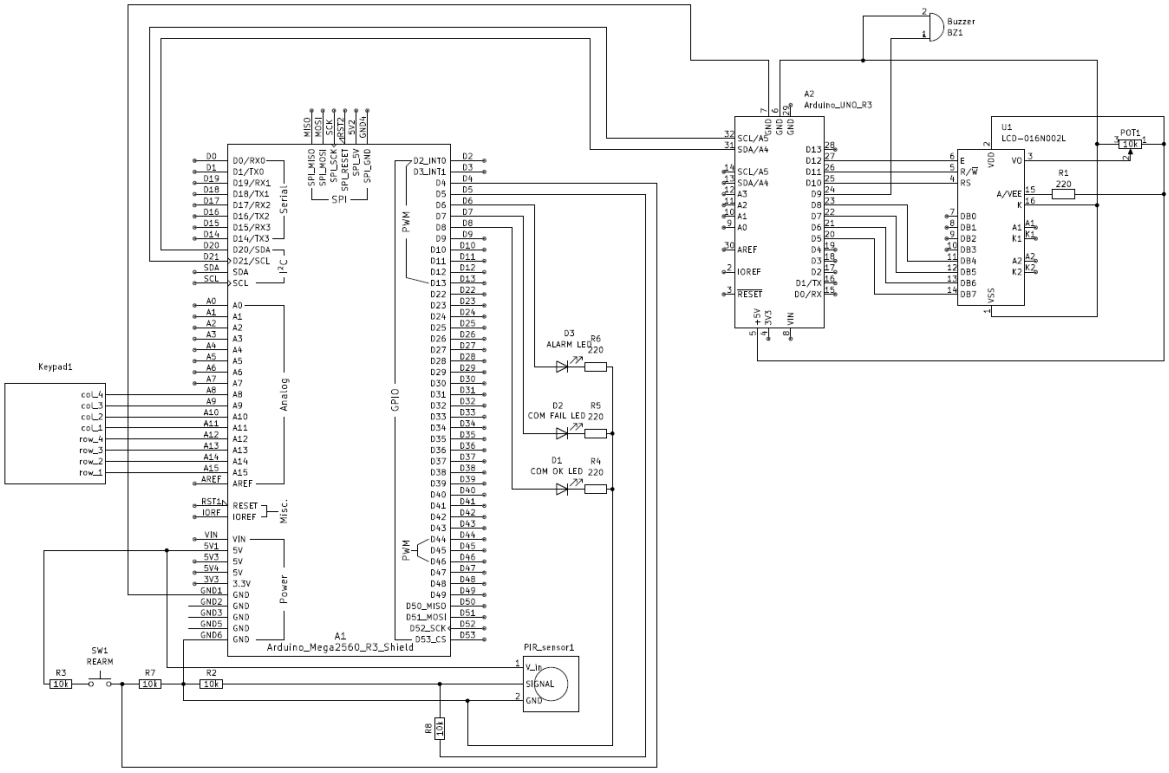

Figure 1.1 Circuit diagram

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

3.2.1 Arduino Uno

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

As Uno receives state change information or password information it displays it through LCD screen. When it receives that alarm state is activated it starts the alarm which is played via the buzzer. As Mega moves to next state and Uno receives information of that, it stops the alarm.

3.2.2 Arduino mega

Arduino Mega is connected to Uno, three LEDs, motion detector, Rearm and to keypad. This is because Mega was chosen to be receiver and user of user input. User input is received through 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 debug and indicate to user 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 user that mega is in alarm state to know that Mega and Uno are cooperating.

In the demonstration video, viewer can observer that the use of rearm button had some problems which probably where with the connection between breadboard and wires or between breadboard and legs of the button. This resulted in not always using the button of wire from 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 a one feature changed from the instructions.

4.1 Mistakes

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

4.2 Unimplemented features

We had some more ideas what could be implemented to the system but as 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 user inputted correct password, Uno would play part of the song Bad Apple.
Displaying passwords	Uno would have displayed 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 light up the current one. Unnecessary as LCD screen does the same.
Password change state	A state where user could change password even without EEPROM.

4.3 Changed feature.

We changed the rearm to only be able to be activated when 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 alarm by rearming could be used as loophole in fault injection attack by intruder as inputting current in specific wire would disable alarm.

4.4 Improvements

There are some obvious improvements to the system like rest of the additional functionalities or the features mentioned in the unimplemented list as more features there is also upgrade to the current system.

- Uno could have the buzzer be interrupt driven so playing changing sound would been possible while operating others normally.
- Inputting wrong password doesn't cause timer to stop which might lead to times up. This doesn't cause problems but causes uno to display time's up text on LCD.
- The alarm is only one annoying note which could be replaced with another song.
- If 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 same problems as with the weekly exercises. We made most of the additional functionalities and made three own functionalities, although small but handy. During the exercise we learned to use I2C properly and designing 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