# CAB202 Assignment:

Student Name: Thomas Rowen

Student Number: n9968075

TinkerCAD Link: Couldn’t share as discussed on slack. Here is my project link I will use to open my project and sign in during the demo: [https://www.tinkercad.com/things/aMIMxO7To8h-cab202-assignment-tom-rowen/](https://www.tinkercad.com/things/aMIMxO7To8h-cab202-assignment-tom-rowen/editel)

Or

https://www.tinkercad.com/things/aMIMxO7To8h-cab202-assignment-tom-rowen/editel

## Introduction

*Functionality Implemented*

*Note to Teacher: Extended Functionality is* ***Bolded*** *and includes:*

* *Keyboard switch matrix (2 points)*
* *Using Sensors/actuators (2 points)*
* *Internal pull-up resistors (1 Point)*
* *Pin Change interrupts (1 point)*

*Case:*

*Recently my parents shed was robbed on our property. Afterwards we were quoted $8000 for a security system. The value proposition of my project is to provide a cheap security alternative to my father by making use of sensors as apposed to cameras. These may be added later when I physically implement my solution. This solution is intended to be a working prototype.*

|  |  |
| --- | --- |
| Digital I/O - Switch | *To fulfill this functionality, I implemented a* ***keyboard switch matrix****. This satisfies the criteria as looking at line 133 to 147 and the related schematic. It can be that column pins are set to* ***input pullups internally*** *and then rows are set to output. This corresponds to my readkey() function on line 374. Where a looping function mechanism changes row output from high to low one pin at a time. This means as all times row output pins will have 3 pins set high and one pin set low. When a subsequent column is pressed, the column input signal will change from high (****pullup****) to low. This I/O pin voltage logic allows the program to implement logic to determine which key has been pressed in the keyboard 4x4 matrix.*  *This feature is integral to the success of my application. It allows the system to be secured/deactivated using a unique password. Known only by the user. For instance, if a burglary takes place. The alarm will be unable to be turned off without password knowledge. This is the same for activating the alarm. Additionally, the 4x4 matrix made more pins available for other features needed in the system such as a* ***PIR sensor****. Which also provides digital input into the program.* |
| Digital I/O – Debouncing | *(Maybe implemented)*  *Within the same readkey() function on line 374 it can be seen that once a column pin has registered a press and subsequently been set to low. It will hold the program while this column input pulldown key is being held down – this logic is captured in the while statement below each if statement for column pin logic. This allows buttons to be held down without the program registering multiple clicks. This adds value to the application as an effective keypad is essential for an effective alarm system. For instance, if debouncing is not implement. The product user may find it difficult to activate/deactivate the alarm in a timely manner and may result in the alarm-speaker being accidentally sounded through faulty password deactivation (false positive).* |
| Digital I/O – LED | *My LED allows the state of the alarm system to be viewed through an LED, as opposed to the LCD and serial ports. For instance, when the system is armed but not activated (through motion/temperature). The red LED is on. However, once motion is triggered and the alarm is activated. The red LED will begin to flash. Showing the system user that the alarm has been triggered and must be deactivated. This will also help with debugging once I implement this application in my father’s shed. For instance, if the LED is flashing but no alarm is sounding. I know the issue is with the speaker system in place or the duty cycle applied to it once motion is triggered (refer to* ***Pin-Change ISR line 164*** *and TIMER2 ISR line 174).*  *This feature can be changed once physical implementation takes place to simulate a strobe light. This will increase the deterrence of burglars, increasing system effectiveness.* |
| Analog Input – ADC | *In a fire situation. The product user wishes to have their alarm system sounded so that early prevention measures can be implemented (eg calling fire brigade). For this, TIMER2 checks the adc value of the* ***temperature sensor*** *each time the timer overflows. If this value is over that of the corresponding Celsius value of 100 degrees. The alarm will trip, shown on line 217. This feature works in all program states. For instance, if the alarm is not armed but rather in idle state- the alarm will still trigger in a fire situation.* |
| Analog Output – PWM | *An effective alarm system requires an effective deterrent such as an alarm. To increase the deterrence effectiveness of the alarm. A software PWM on/off feature was implemented. This will be more effective in deterring robbers as a constantly changing alarm sound (through duty cycle change) will create a more panicked environment for potential thieves. Much like the LED strobe feature discussed in the LED section. The duty cycle is also effective in controlling the on-off state of the alarm.* |
| Serial I/O – UART | *Output: Data relating to the program status and necessary tasks to be performed by the user such as enter password are outputted into the serial console. This provided value to my overall application by creating a testing framework that could be used to develop the remaining application. Furthermore, in the event of the LCD failing. The serial output can be used as a backup to view system information.*  *Input: Not implemented* |
| LCD | *Much like serial output, the lcd provides the user with a method of viewing program states and necessary tasks to perform. Such as entering and setting up the password. If this were not implemented. The user may enter their password to deactivate the alarm before the program is ready to receive characters, resulting in false positive alarm activation.* |
| Timers (other than debouncing or PWM) | *ISR(TIMER2\_OVF\_vect) was one of the main and most important features of my application. Other than PWM and debouncing, this timer controls the status checking of the* ***thermometer sensor*** *ADC value and LED flashing. The timer also controls when the alarm is sounded. This an important feature as consistent checking of* ***sensor values*** *is essential to any effective microcontroller alarm system. See line 174 for timer implementation.* |
| Extended functionality | *Other than the extended functionality discussed above in* ***Bold.*** *My system also effectively uses a* ***motion sensor*** *with a* ***pin-change interrupt.*** *A motion senser provides essential digital input the program uses to determine whether the alarm should be sounded. My program uses an* ***internal pull-down resister*** *as configured on line 117. This allows the rising (and soon-after falling) edge to be detected by the pin-change interrupt on line 164. Which subsequently changes the state and duty cycle controlling the buzzer in TIMER2.* |

## Schematic

Please zoom in for clear view.

*Diagram, schematic

Description automatically generated*

## Wiring Instructions

|  |  |  |
| --- | --- | --- |
| **Step** | **Task** | **Instruction** |
| 1 | Place Arduino | Place Arduino uno R3 in centre of workspace with Digital PWM pins facing upwards. Power and analogue pins should be facing downwards. |
| 2 | Place Keypad | Place keypad above the Arduino so that the keypad pins are approximately above digital pins 13 to 6. |
| 3 | Connect Keypad | Connect row 1 keypad pin to digital I/O pin 2 (D2). Connect row 2 keypad pin to digital I/O pin 12 (D12). Connect row 3 keypad pin to digital I/O pin 11 (D11). Connect row 4 keypad pin to digital I/O pin 10 (D10). Connect column 1 keypad pin to digital I/O pin 9 (D9). Connect column 2 keypad pin to digital I/O pin 8 (D8). Connect column 3 keypad pin to digital I/O pin 7 (D7). Connect column 4 keypad pin to digital I/O pin 6 (D6). |
| 4 | Connect TMP36 Sensor | Place TMP36 temperature sensor to the bottom left of the Arduino, leaving space below analog and power Arduino pins free. Connect power pin of TMP36 to 5v Arduino power pin. Connect TMP36 Vout pin to Arduino analog in pin A0. Lastly, connect TMP36 GND pin to the Arduino GND pin at the bottom of the board |
| 5 | Place Breadboard | Place small breadboard to the right of Arduino with some space in between the two components. Vout and GND horizontal breadboard lines should be at the top and bottom of breadboard placement. |
| 6 | Place LCD | Place LCD below both he Arduino and breadboard for wire management. |
| 7 | Place potentiometer | Place Potentiometer with 250 kiloohm resistance to the right of Arduino underneath and slightly to the right of the breadboard |
| 8 | Connect LCD | Starting from the leftmost LCD pin GND/Ground. Connect this LCD pin to the respective GND pin at the bottom of the Arduino. Connect the LCD VCC/power pin to Arduino 5v pin. Connect the LCD V0/Contrast pin to the wiper (centre) pin of the potentiometer. Next, connect the LCD RS/register select pin to the Arduino digital I/O pin 4 (D4). Connect the RW/Read-write LCD pin to the breadboard’s horizontal ‘+’ line, this will later be connected to ground. The same can be done for the LED cathode LCD pin. Next, connect LCD E/Enable pin to the Arduino A1 pin. Do not connect LCD pins DB0-DB3. LDC pins DB4, DB5, DB6 and DB7 should be connected to the respective Arduino analog input pins A2, A3, A4 and A5. Connect LCD LED/LED cathode pin to the vertical breadboard line at coordinates (f,3). |
| 9 | Place and connect resistor | Place a 1000 ohms resistor on the breadboard so that resistor terminal 1 is connected to vertical breadboard line at coordinates (j,3). Terminal 1 should therefore be connected to LCD LED cathode pin. Terminal 2 connects to horizontal breadboard line ‘-‘. |
| 10 | Connect Potentiometer | Potentiometer wiper pin should already be connected to LCD V0 as per step 8. Continue to connect potentiometer terminal 1 pin to breadboard horizontal line ‘-‘ and potentiometer terminal 2 to breadboard horizontal line ‘+’. |
| 11 | Connect breadboard | Connect the left-most horizontal breadboard pin line labelled ‘-‘ to Arduino 5v/power pin at the bottom of Arduino. Next, connect the left-most horizontal breadboard pin labelled ‘+‘ to Arduino GND/Ground pin at the bottom of Arduino. |
| 12 | Connect LED/Resistor | Place red LED and 1000 ohms resistor in between the Arduino and breadboard. Connect terminal 1 of the resistor to the LED Anode Node. Connect resistor terminal 2 to Arduino Digital I/O pin 3 (D3). Last, connect the LED Cathode pin to breadboard horizontal pin ‘+’. |
| 13 | Connect PIR Sensor | Place PIR motion sensor above all system components. Connect PIR sensor signal pin (pin 0) to Arduino digital I/O pin 13. Next, connect PIR sensor power pin (pin 1) to the horizontal breadboard line ‘-‘. Lastly, connect the PIR sensor ground pin (pin 2) to horizontal breadboard line ‘+’. |
| 14 | Place/connect Piezo and resistor | Place piezo speaker above breadboard and to the right of PIR sensor ground and power connections. Place a 1000 ohms resistor so that the resistor’s terminal 2 is **directly** connected to piezo negative pin. Connect the resistor’s terminal 1 pin to the Arduino’s digital I/O pin 5 (D5). Connect the remaining piezo positive pin to horizontal breadboard line ‘+’. |
| 15 | Place Oscilloscope  *(For testing)* | To test PWM, place an oscilloscope above the piezo and below the PIR sensor. Next connect the negative oscilloscope pin to the negative piezo pin or terminal 2 of resistor 4 – as it is directly connected (see step 14). Lastly, connect the positive oscilloscope pin to the positive piezo pin. |

Text

Description automatically generated