PIC PROJECT - TITLE

1) Design Brief

MULTI-FUNCTIONAL CAR SECURITY

In the Local area in Blackpool, the amount of crimes surrounding motor vehicles has risen by a large amount. (43% precisely) My aim is to create a multi-functional car system which aims to mitigate the amount of vehicle related crimes.

Security is a large issue when it comes to vehicles, there are many systems in place to protect the user from physical damage, such as crashing incidents but there is not much security in place to ensure that customers have easy access to emergency services and to stop their vehicles from being taken in a worst-case scenario.

Security systems need to be developed well as customers are interested in staying safe while driving, and I am sure to make a system which will give drivers the safety and security they desire.

2) Specification

> When a locked car door is opened and the door sensor is activated, sound an alarm after 0.5 seconds, for 30 second intervals with a 7 second wait time in between intervals. (With a 2% tolerance)

> If a car is locked, and force is used to enter the vehicle (e.g. Breaking a window) and the shock/impact sensors activate, sound an alarm after 0.5 seconds for 30 second intervals with a 7 second wait time in between intervals. (With a 2% tolerance)

> If a specific button is pressed, dial the nearest emergency services to dispatch help to the location of the vehicle.

> When the interrupt button is pressed, stop all other actions and lock all actions of the vehicle to ensure that the car cannot be used if someone tries to take the vehicle.

> When the reset button is pressed, end the emergency services call in 0.2 seconds (2% tolerance) in the case of a false alarm.

> When the reset button is pressed and held down, unlock all vehicle actions after 5 seconds (5% tolerance) in the case of a false alarm.

> After a press of a button, lock all car doors and raise windows in 1 second. (5% tolerance)

> Reset button to unlock all doors and windows.

> When reset button is pressed, turn alarm off after 0.2 seconds (2% Tolerance)

3) Inputs/Outputs

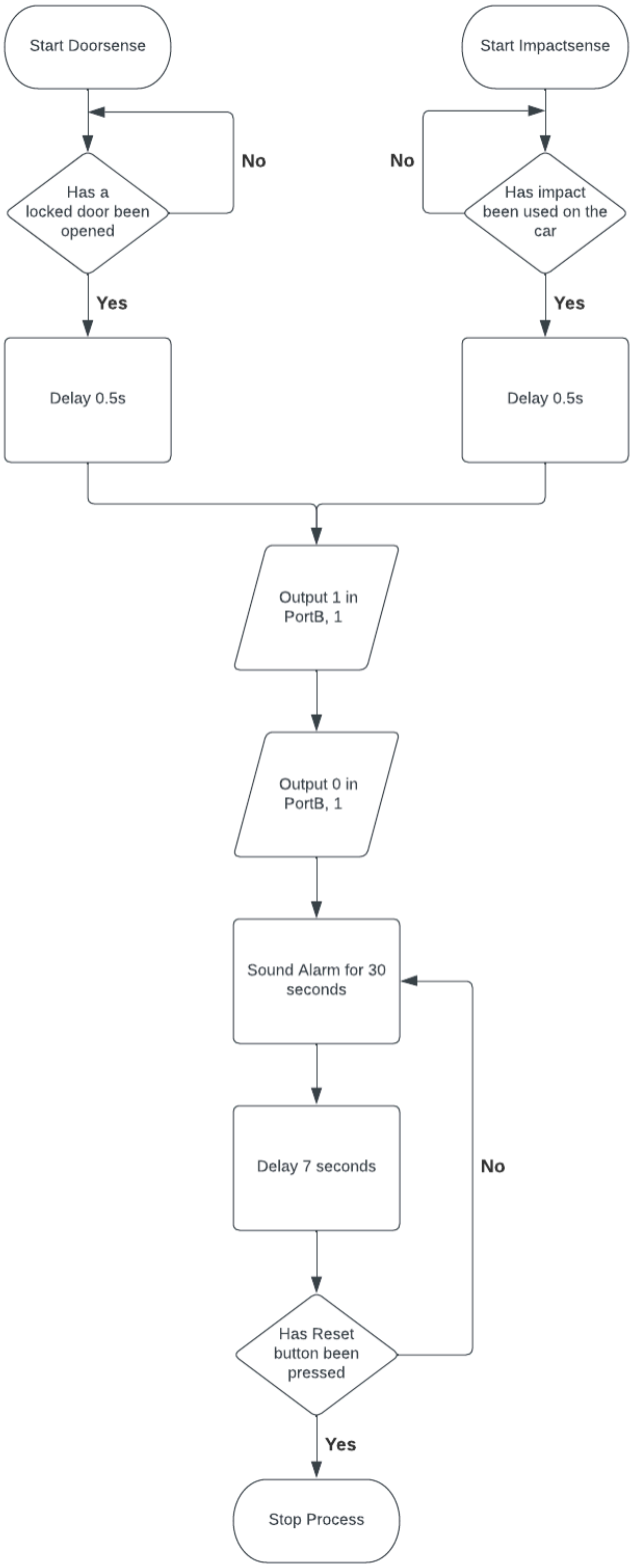
PORT A	INPUTS
PORT A, 0	Reset Button
PORT A, 1	Door sensors
PORT A, 2	Shock/Impact sensors
PORT A, 3	Emergency Services dial button

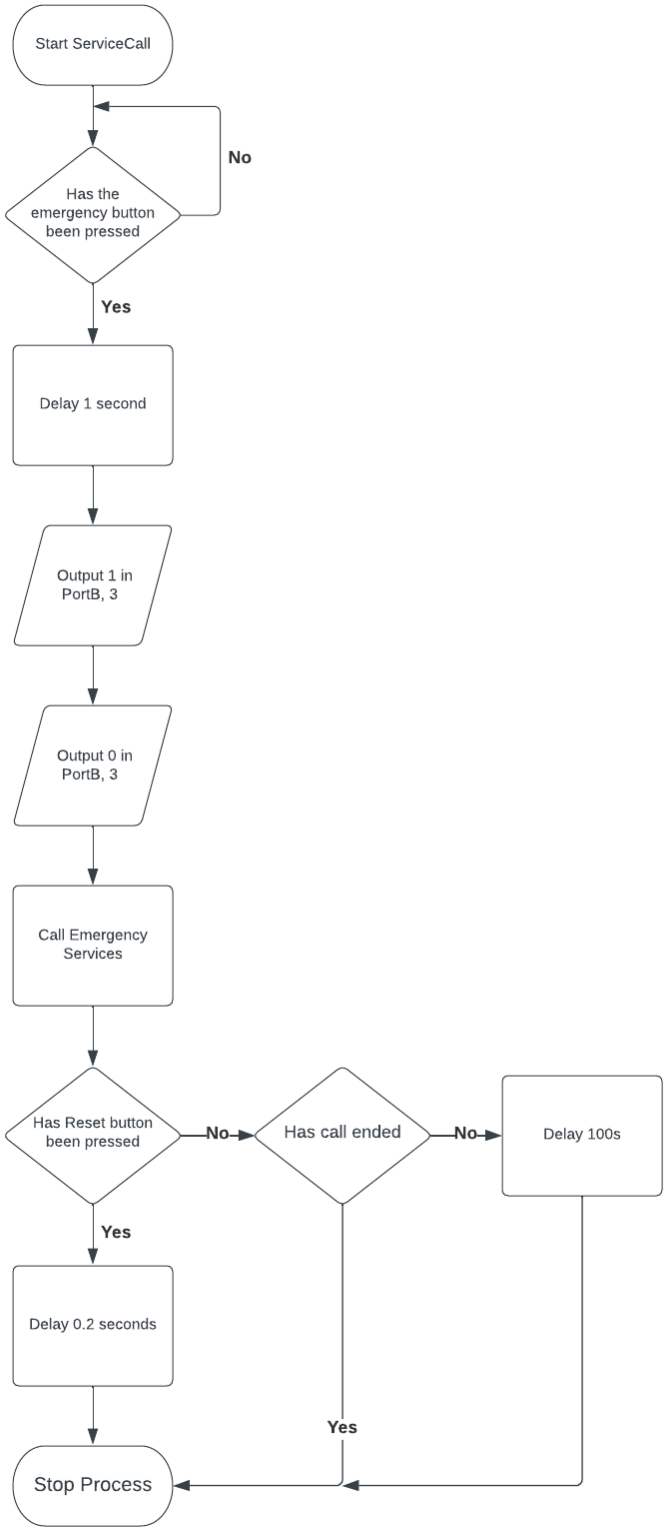
PORT A, 4	Lock doors and windows button
PORT A, 5	Not in use
PORT A, 6	Not in use
PORT A, 7	Not in use

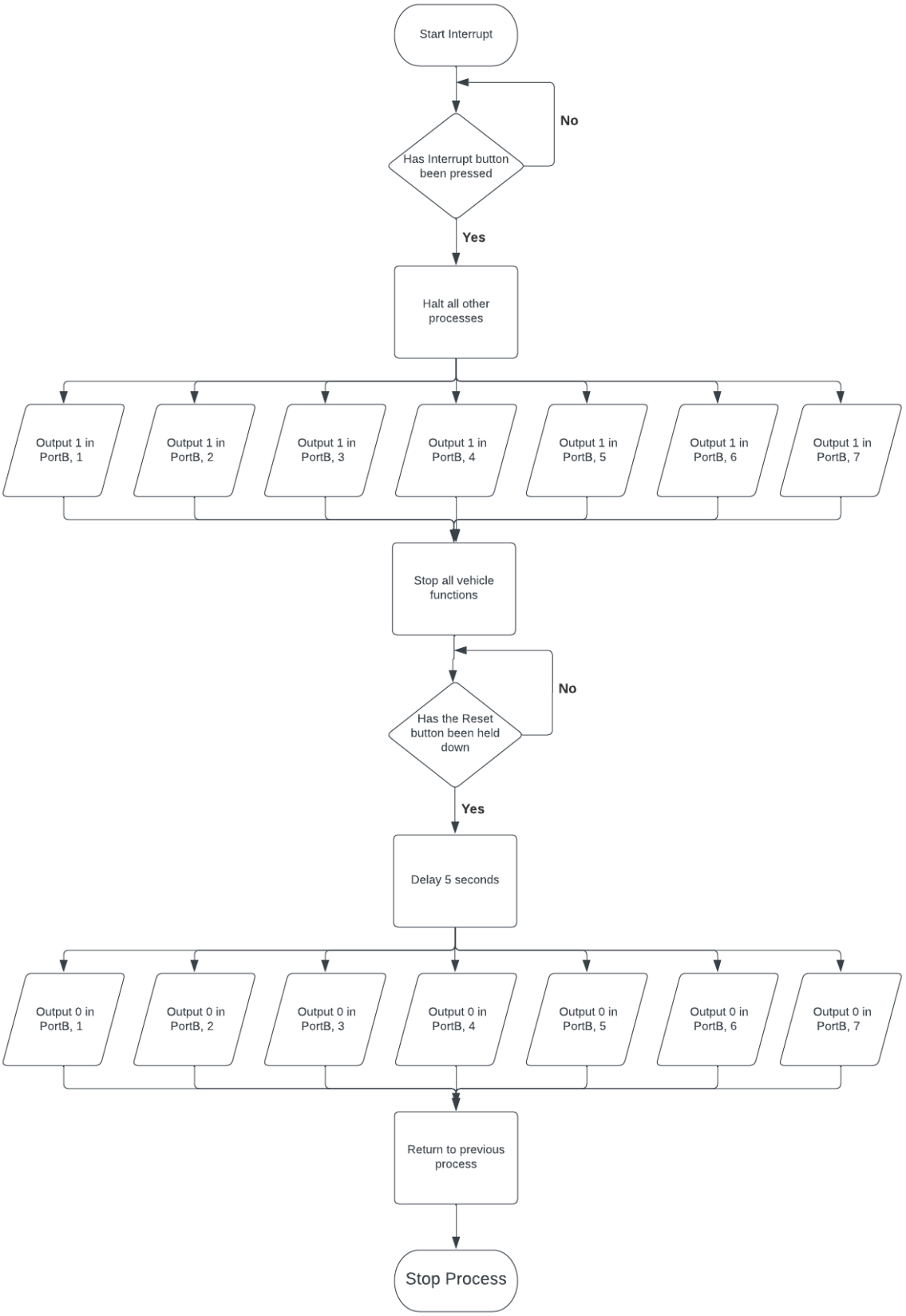
PORT B	OUTPUTS
PORT B, 0	Interrupt button
PORT B, 1	Flash when sensors have been activated / Turn on when interrupt button pressed
PORT B, 2	Flash when calling emergency services / Turn on when interrupt button pressed
PORT B, 3	Flash when calling emergency services / Turn on when interrupt button pressed
PORT B, 4	Flash when calling emergency services / Turn on when interrupt button pressed
PORT B, 5	Flash when calling emergency services / Turn on when interrupt button pressed
PORT B, 6	Turn on when interrupt button pressed / Flash when doors are locked

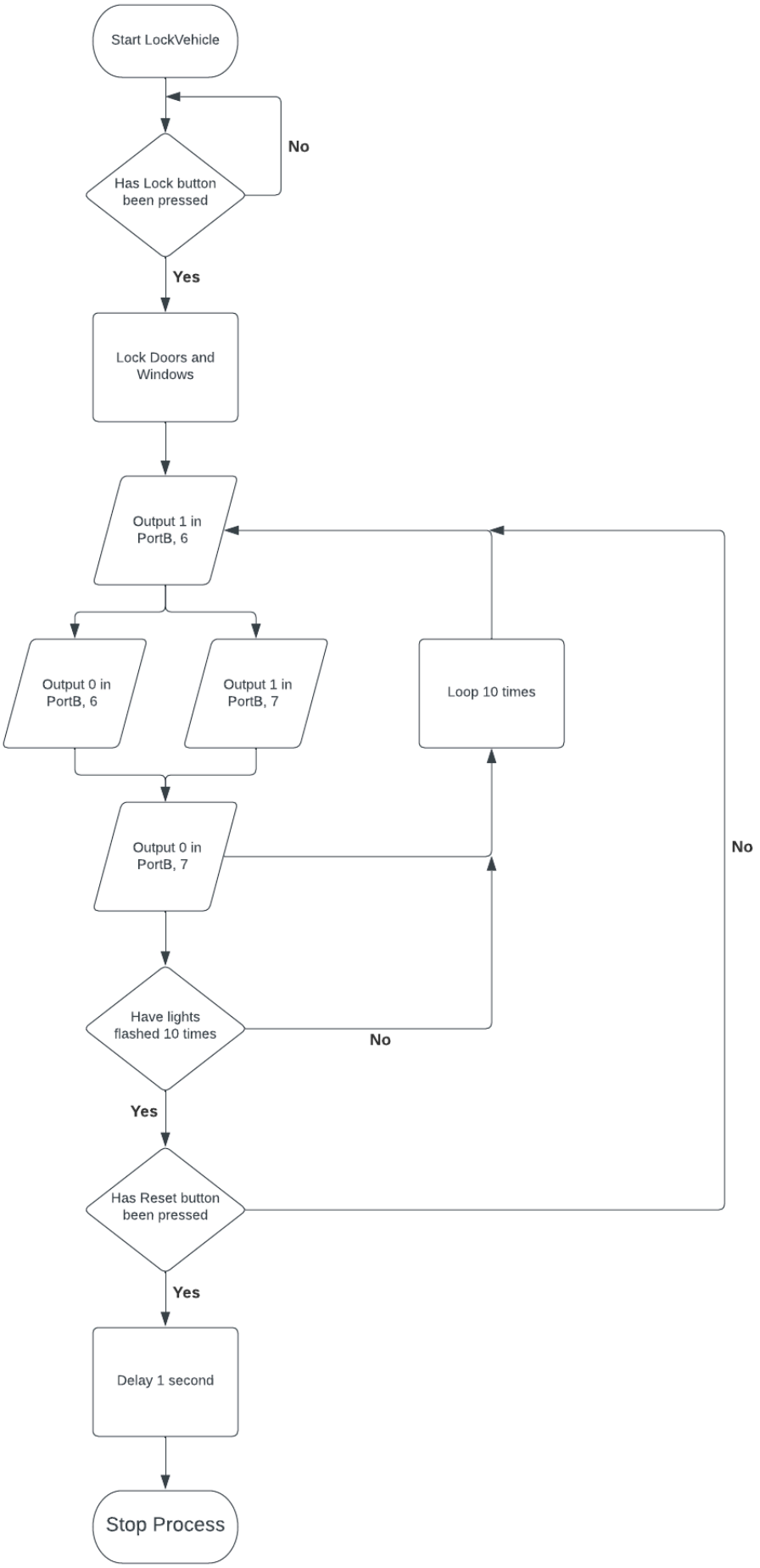
PORT B, 7	Turn on when interrupt button pressed / Flash when doors are locked

4) Flow chart


Start Doorsense
Start Impactsense
Has a locked door been opened
No
Yes
Has impact been used on the car
No
Yes
Delay 0.5s
Delay 0.5s
Output 1 in PortB, 1
Output 0 in PortB, 1
Sound Alarm for 30 seconds
Delay 7 seconds
Has Reset button been pressed
No
Yes
Stop Process


Start ServiceCall
Has the emergency button been pressed
No
Yes
Delay 1 second
Output 1 in PortB, 3
Output 0 in PortB, 3
Call Emergency Services
Has Reset button been pressed
No
Has call ended
No
Delay 100s
Yes
Delay 0.2 seconds
Yes
Stop Process


Start Interrupt
Has Interrupt button been pressed
No
Yes
Halt all other processes
Output 1 in PortB, 1
Output 1 in PortB, 2
Output 1 in PortB, 3
Output 1 in PortB, 4
Output 1 in PortB, 5
Output 1 in PortB, 6
Output 1 in PortB, 7
Stop all vehicle functions
Has the Reset button been held down
No
Yes
Delay 5 seconds
Output 0 in PortB, 1
Output 0 in PortB, 2
Output 0 in PortB, 3
Output 0 in PortB, 4
Output 0 in PortB, 5
Output 0 in PortB, 6
Output 0 in PortB, 7
Return to previous process
Stop Process


Start LockVehicle
Has Lock button been pressed
No
Yes
Lock Doors and Windows
Output 1 in PortB, 6
Output 0 in PortB, 6
Output 1 in PortB, 7
Loop 10 times
Output 0 in PortB, 7
No
Have lights flashed 10 times
No
Yes
Has Reset button been pressed
Yes
Delay 1 second
Stop Process

5) Program

```
;****** PROGRAM ******
;************* remove semicolons from next two lines to enable interrupt routine************
bsf INTCON,INT0IE                ;set external interrupt enable
bsf INTCON,GIE                   ;enable all interrupts
main
;*********************************************************************

BEGIN
        clrf PORTB
        btfsc PORTA,0            ;if PORTA,0 is pressed, then GOTO STOP. if not pressed, skip
        GOTO STOP
        btfsc PORTA,1            ;if PORTA,1 is pressed, then GOTO DOORSENSE. if not pressed, skip
        GOTO DOORSENSE
        btfsc PORTA,2            ;if PORTA,2 is pressed, then GOTO IMPACTSENSE. if not pressed, skip
        GOTO IMPACTSENSE
        btfsc PORTA,3            ;if PORTA,3 is pressed, then GOTO SERVICECALL. if not pressed, skip
        GOTO SERVICECALL
        btfsc PORTA,4            ;if PORTA,4 is pressed, then GOTO LOCKVEHICLE. if not pressed, skip
        GOTO LOCKVEHICLE
        GOTO BEGIN               ;creates a loop

STOP
        clrf PORTB               ;clears all PORTB outputs
        GOTO BEGIN               ;loops back to beginning

DOORSENSE                        ;senses if a locked door has been opened
        clrf PORTB               ;clears all PORTB outputs
        call wait100ms           ;waits 0.5s
        call wait100ms
        call wait100ms
        call wait100ms
        call wait100ms
        GOTO ALARM

IMPACTSENSE                      ;senses if forces was used to access vehicle
        clrf PORTB               ;clears all PORTB outputs
        call wait100ms           ;waits 0.5s
        call wait100ms
        call wait100ms
        call wait100ms
        call wait100ms
        GOTO ALARM
```

```
ALARM
        clrf PORTB              ;clears all PORTB outputs
        movlw d'30'             ;adds 30 to working register
        movwf TIMER             ;names register TIMER
COUNT   bsf PORTB,1             ;outputs 1 from PORTB,1
        call wait100ms          ;waits 0.5s
        call wait100ms
        call wait100ms
        call wait100ms
        call wait100ms
        bcf PORTB,1             ;outputs 0 from PORTB,1
        call wait100ms          ;waits 0.5s
        call wait100ms
        call wait100ms
        call wait100ms
        call wait100ms
        btfsc PORTA,0           ;if PORTA,0 is pressed, GOTO BEGIN. if not pressed, skip
        GOTO BEGIN
        decfsz TIMER,F          ;decrements TIMER until 0
        GOTO COUNT              ;if not 0, GOTO COUNT and repeat loop
        GOTO ALARMINT           ;if 0, GOTO ALARMINT

ALARMINT
        btfsc PORTA,0           ;if PORTA,0 is pressed, GOTO BEGIN. if not pressed, skip
        GOTO BEGIN
        call wait1000ms         ;waits 7 seconds
        call wait1000ms
        call wait1000ms
        call wait1000ms
        call wait1000ms
        call wait1000ms
        call wait1000ms
        GOTO ALARM              ;loops back to ALARM

SERVICECALL                     ;clears all PORTB outputs
        clrf PORTB
        GOTO CALLLOOP

CALL1
        movlw d'100'            ;keeps call active for 100 seconds then ends
        movwf CALLDELAY
CALL2   bsf PORTB,3             ;outputs 1 into PORTB,3
        call wait100ms          ;waits 0.5s
        call wait100ms
        call wait100ms
        call wait100ms
        call wait100ms
        bcf PORTB,3             ;outputs 0 into PORTB,3
        call wait100ms          ;waits 0.5s
        call wait100ms
        call wait100ms
        call wait100ms
        call wait100ms
        btfsc PORTA,0           ;if PORTA,0 is pressed, GOTO BEGIN. if not pressed, skip
        GOTO BEGIN
        decfsz CALLDELAY,F      ;decrements CALLDELAY until 0
        GOTO CALL2              ;if not, 0 GOTO CALL2
        GOTO BEGIN              ;if 0, GOTO BEGIN
```

```
CALLLOOP
        call wait100ms          ;waits 0.3s
        call wait100ms
        call wait100ms
        bsf PORTB,2
        call wait100ms          ;waits 0.3s
        call wait100ms
        call wait100ms
        bcf PORTB,2
        bsf PORTB,3
        call wait100ms          ;waits 0.3s
        call wait100ms
        call wait100ms
        bcf PORTB,3
        bsf PORTB,4
        call wait100ms          ;waits 0.3s
        call wait100ms
        call wait100ms
        bcf PORTB,4
        bsf PORTB,5
        call wait100ms          ;waits 0.3s
        call wait100ms
        call wait100ms
        bcf PORTB,5
        bsf PORTB,2
        call wait100ms          ;waits 0.3s
        call wait100ms
        call wait100ms
        bcf PORTB,2
        bsf PORTB,3
        call wait100ms          ;waits 0.3s
        call wait100ms
        call wait100ms
        bcf PORTB,3
        bsf PORTB,4
        call wait100ms          ;waits 0.3s
        call wait100ms
        call wait100ms
        bcf PORTB,4
        bsf PORTB,5
        call wait100ms          ;waits 0.3s
        call wait100ms
        call wait100ms
        bcf PORTB,5
        bsf PORTB,2
```

```
call wait100ms          ;waits 0.3s
call wait100ms
call wait100ms
bcf PORTB,2
bsf PORTB,3
call wait100ms          ;waits 0.3s
call wait100ms
call wait100ms
bcf PORTB,3
bsf PORTB,4
call wait100ms          ;waits 0.3s
call wait100ms
call wait100ms
bcf PORTB,4
bsf PORTB,5
call wait100ms          ;waits 0.3s
call wait100ms
call wait100ms
bcf PORTB,5
GOTO CALL1
```

```
LOCKVEHICLE                     ;locks vehicle doors and windows
        clrf PORTB              ;clears all PORTB outputs
        call wait100ms          ;waits 0.1 seconds
        GOTO PATTERN

PATTERN                         ;PATTERN only stops if reset button is pressed
        btfsc PORTA,0           ;if PORTA,0 is pressed, GOTO BEGIN. if not pressed, skip
        GOTO BEGIN
        bsf PORTB,6             ;outputs 1 into PORTB,6
        call wait100ms          ;waits 0.1s
        bcf PORTB,6             ;outputs 0 into PORTB,6
        bsf PORTB,7             ;outputs 1 into PORTB,7
        call wait100ms          ;waits 0.1s
        bcf PORTB,7             ;outputs 0 into PORTB,7
        GOTO PATTERN            ;if not 0, repeat PATTERN

ALARM
        clrf PORTB              ;clears all PORTB outputs
        movlw d'30'             ;adds 30 to working register
        movwf TIMER             ;names register TIMER
COUNT   bsf PORTB,1             ;outputs 1 from PORTB,1
        call wait100ms          ;waits 0.5s
        call wait100ms
        call wait100ms
        call wait100ms
        call wait100ms
        bcf PORTB,1             ;outputs 0 from PORTB,1
        call wait100ms          ;waits 0.5s
        call wait100ms
        call wait100ms
        call wait100ms
        call wait100ms
        btfsc PORTA,0           ;if PORTA,0 is pressed, GOTO BEGIN. if not pressed, skip
        GOTO BEGIN
        decfsz TIMER,F          ;decrements TIMER until 0
        GOTO COUNT              ;if not 0, GOTO COUNT and repeat loop
        GOTO ALARMINT           ;if 0, GOTO ALARMINT

ALARMINT
        btfsc PORTA,0           ;if PORTA,0 is pressed, GOTO BEGIN. if not pressed, skip
        GOTO BEGIN
        call wait1000ms         ;waits 7 seconds
        call wait1000ms
        call wait1000ms
        call wait1000ms
        call wait1000ms
        call wait1000ms
        call wait1000ms
        GOTO ALARM              ;loops back to ALARM

RESET
        clrf PORTB
        GOTO BEGIN
```

```
;*********************************************************************
; INTERRUPT SERVICE ROUTINE
;*********************************************************************

W_SAVE      EQU B2              ; backup registers used in interrupts
interrupt
            movwf W_SAVE        ; Copy W to save register
            btfss INTCON,INT0IF ; check correct interrupt has occurred
            retfie              ; no, so return and re-enable GIE
            GOTO LOOP1
;**********The interrupt service routine (if required) goes here*********
LOOP1       call wait1000ms
            bsf PORTB,1
            bsf PORTB,2
            bsf PORTB,3
            bsf PORTB,4
            bsf PORTB,5
            bsf PORTB,6
            bsf PORTB,7
            btfss PORTA,0
            GOTO LOOP1
            btfsc PORTA,0
            call wait1000ms
            btfsc PORTA,0
            call wait1000ms
            btfsc PORTA,0
            call wait1000ms
            btfsc PORTA,0
            call wait1000ms
            btfsc PORTA,0
            call wait1000ms
            btfss PORTA,0
            GOTO LOOP1
            clrf PORTB
            bcf INTCON,INT0IF   ; clear interrupt flag
            movf W_SAVE,W       ; restore W
            retfie              ; return and re-set GIE bit
            END                 ; all programs must end with this
```

6) Evidence of Testing

Process of testing:

When I am to test the PIC chip to ensure that all features are working as intended, there is a short process to work through in order for the computer to be able to read the commands. After inserting the assembly source file into the MPASM program. This assembles the file and turns the code into machine readable code which the computer can recognise. After this, I need to insert the HEX file into MLoader which will send and execute the file, and allow for the PIC to function. If there are any errors in the programming, MPASM will have picked up on this and I will be met with an error, any error which I encounter will be described below.

Test 1 — Error due to parameters

```
6   Error[113]   I:\16F88TEMPLATE2021.ASM 154 : Symbol not previously defined (HALT)
```

Before:

```
148 main
149 ;*********************************************************************
150
151 BEGIN
152         clrf PORTB
153         BTFSC PORTA,0           ;check porta,0 if zero skip next line
154         GOTO HALT
155         BTFSC PORTA,1           ;check porta,1 if zero skip next line
156         GOTO DOORSENSE
157         BTFSC PORTA,2           ;check porta,2 if zero skip next line
158         GOTO IMPACTSENSE
159         BTFSC PORTA,3           ;check porta,3 if zero skip next line
160         GOTO SERVICECALL
161         BTFSC PORTA,4           ;check porta,4 if zero skip next line
162         GOTO LOCKVEHICLE
163         GOTO BEGIN
164 DOORSENSE
165         clrf PORTB
166         bsf PORTB,0
167         bcf PORTB,0
168         GOTO BEGIN
169
170 IMPACTSENSE
171         clrf PORTB
172         bsf PORTB,1
173         bcf PORTB,1
174         GOTO BEGIN
175
176 SERVICECALL
177         clrf PORTB
178         bsf PORTB,2
179         bcf PORTB,2
180         GOTO BEGIN
181
182 LOCKVEHICLE
183         clrf PORTB
184         bsf PORTB,3
185         bcf PORTB,3
186         GOTO BEGIN
187
188 HALT
189         clrf PORTB
190         bsf PORTB,4
191         bcf PORTB,4
192         GOTO BEGIN
```

At first, the assembly code here did not assemble correctly into a HEX file due to an error. The error was on line 154 and 188, the use of the word HALT affected the assembly code. After correcting this and changing the word to STOP, the code worked correctly. To test the button's function, I linked each button to a single light output on PORT B. When each input on PORT A was pressed, it lit up the corresponding light on PORT B.

After:

```
main
;*********************************************************************

BEGIN
        clrf PORTB
        BTFSC PORTA,0           ;check porta,0 if zero skip next line
        GOTO STOP
        BTFSC PORTA,1           ;check porta,1 if zero skip next line
        GOTO DOORSENSE
        BTFSC PORTA,2           ;check porta,2 if zero skip next line
        GOTO IMPACTSENSE
        BTFSC PORTA,3           ;check porta,3 if zero skip next line
        GOTO SERVICECALL
        BTFSC PORTA,4           ;check porta,4 if zero skip next line
        GOTO LOCKVEHICLE
        GOTO BEGIN
DOORSENSE
        clrf PORTB
        bsf PORTB,0
        bcf PORTB,0
        GOTO BEGIN

IMPACTSENSE
        clrf PORTB
        bsf PORTB,1
        bcf PORTB,1
        GOTO BEGIN

SERVICECALL
        clrf PORTB
        bsf PORTB,2
        bcf PORTB,2
        GOTO BEGIN

LOCKVEHICLE
        clrf PORTB
        bsf PORTB,3
        bcf PORTB,3
        GOTO BEGIN

STOP
        clrf PORTB
        bsf PORTB,4
        bcf PORTB,4
        GOTO BEGIN
```

Test 2 — Not looping due to misplacement of code

Before:

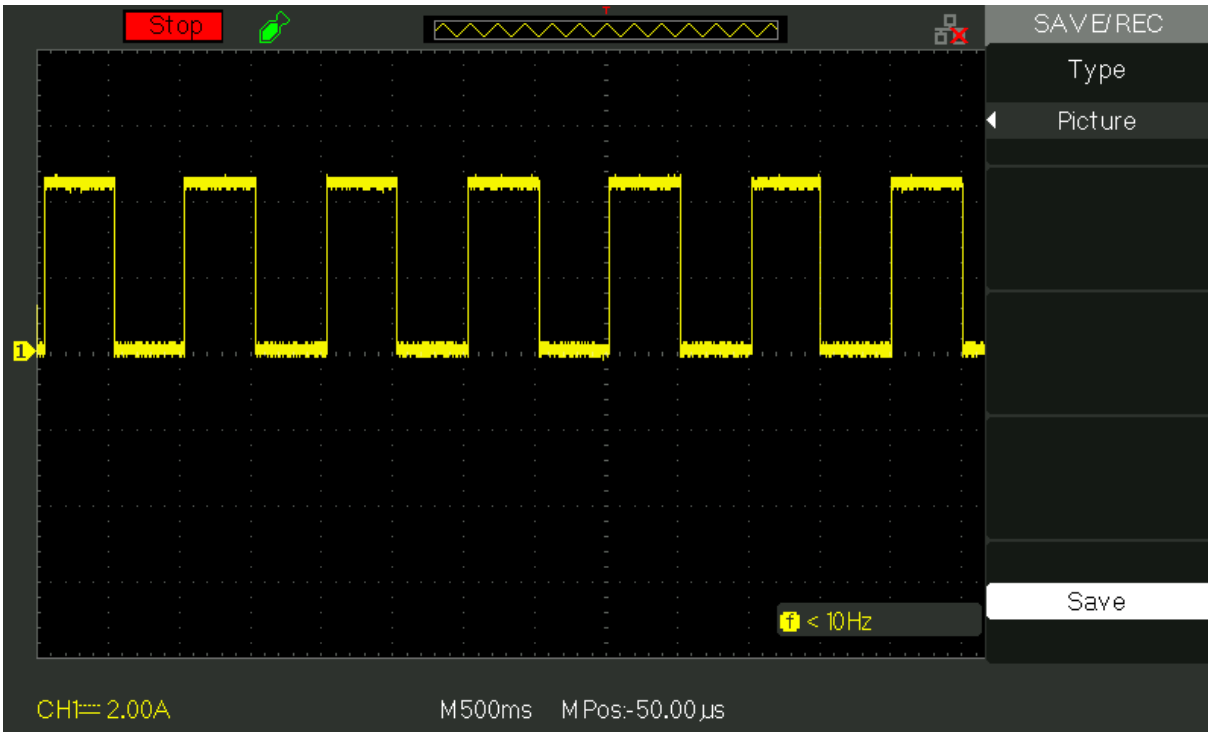
```
ALARM
        clrf PORTB              ;clears all PORTB outputs
        movlw d'10'             ;adds 30 to working register
        movwf TIMER             ;names register TIMER
COUNT   btfsc PORTA,0           ;if PORTA,0 is pressed, GOTO BEGIN. if not pressed, skip
        GOTO BEGIN
        bsf PORTB,1             ;outputs 1 from PORTB,1
        call wait100ms          ;waits 0.5s
        call wait100ms
        call wait100ms
        call wait100ms
        call wait100ms
        bcf PORTB,1             ;outputs 0 from PORTB,1
        call wait100ms          ;waits 0.5s
        call wait100ms
        call wait100ms
        call wait100ms
        call wait100ms
        decfsz TIMER,F          ;decrements TIMER until 0
        GOTO COUNT              ;if not 0, GOTO COUNT and repeat loop
        GOTO ALARMINT           ;if 0, GOTO ALARMINT
```

After:

```
ALARM
        clrf PORTB              ;clears all PORTB outputs
        movlw d'10'             ;adds 30 to working register
        movwf TIMER             ;names register TIMER
COUNT   bsf PORTB,1             ;outputs 1 from PORTB,1
        call wait100ms          ;waits 0.5s
        call wait100ms
        call wait100ms
        call wait100ms
        call wait100ms
        bcf PORTB,1             ;outputs 0 from PORTB,1
        call wait100ms          ;waits 0.5s
        call wait100ms
        call wait100ms
        call wait100ms
        call wait100ms
        btfsc PORTA,0           ;if PORTA,0 is pressed, GOTO BEGIN. if not pressed, skip
        GOTO BEGIN
        decfsz TIMER,F          ;decrements TIMER until 0
        GOTO COUNT              ;if not 0, GOTO COUNT and repeat loop
        GOTO ALARMINT           ;if 0, GOTO ALARMINT
```

Due to the placement of [btfsc PORTA,0 GOTO BEGIN] the assembly code was misinterpreted and did not function correctly, after moving this further down the line of the code, it works as intended for multiple functions as this error also occurred elsewhere. When compiling this program, it initially worked, but did not function as intended and did not stop flashing even after the number of 10 had decreased to 0.

Oscilloscope testing:

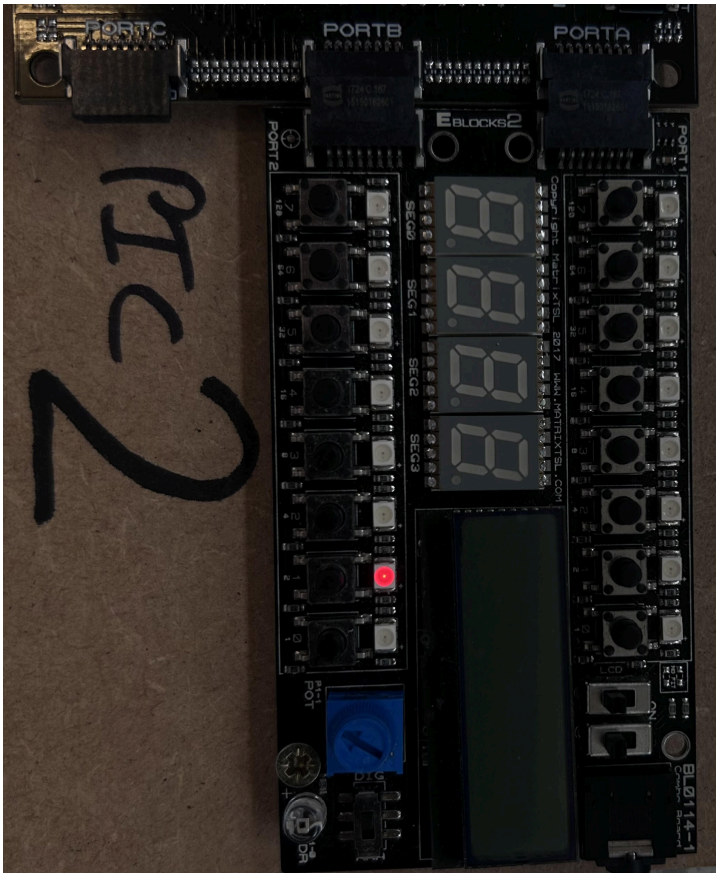


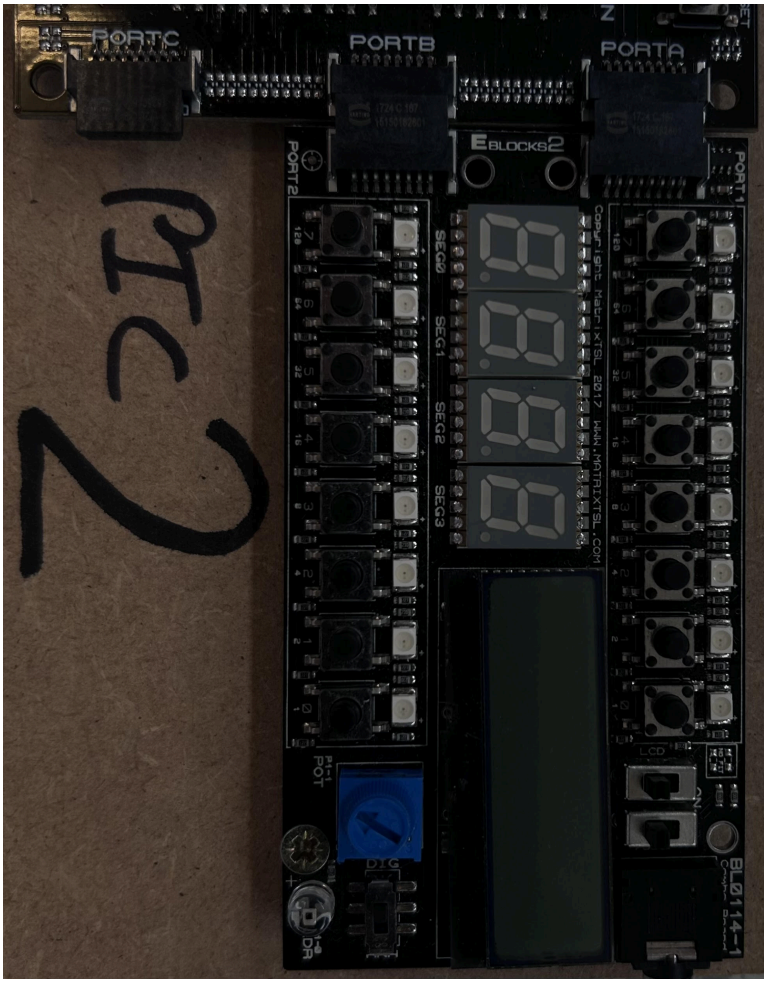
I used an oscilloscope to test the timings and frequencies of the PIC system. When I activated the alarm, I designed it to use 0.5s intervals between each alarm pulse. As seen

above, this has worked. Even with the tolerances in place, I have managed to keep the alarm to staying with 0.5s approximate pulses.

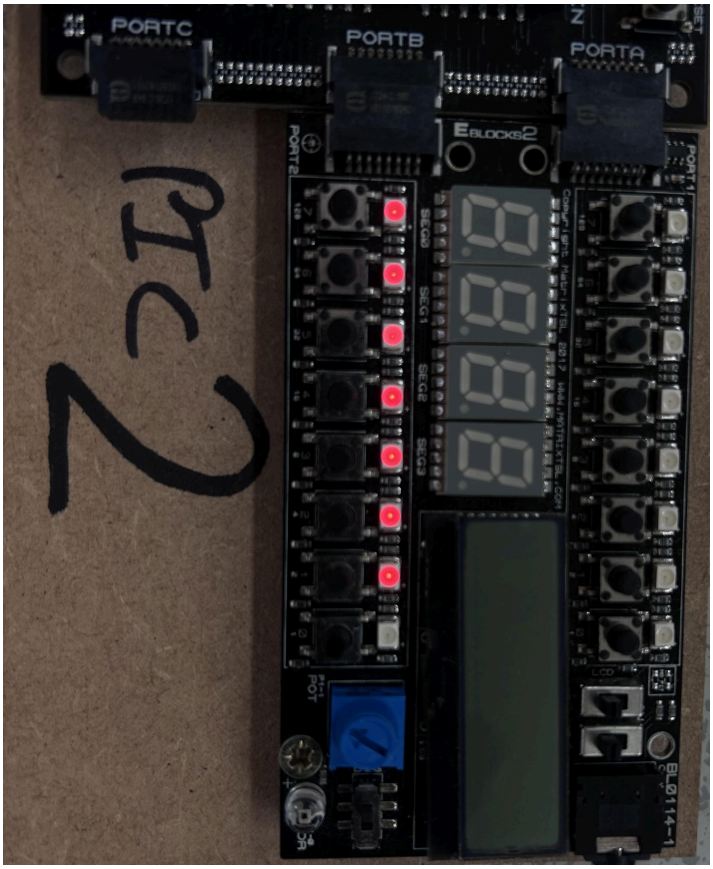
Photos of PIC:

Sensor and alarm:

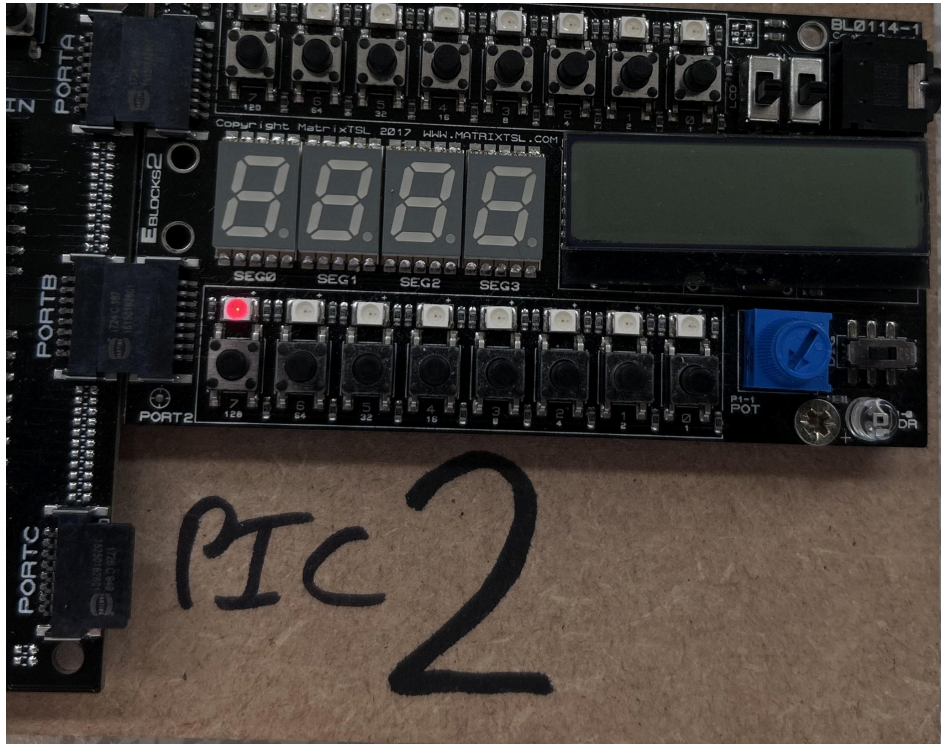


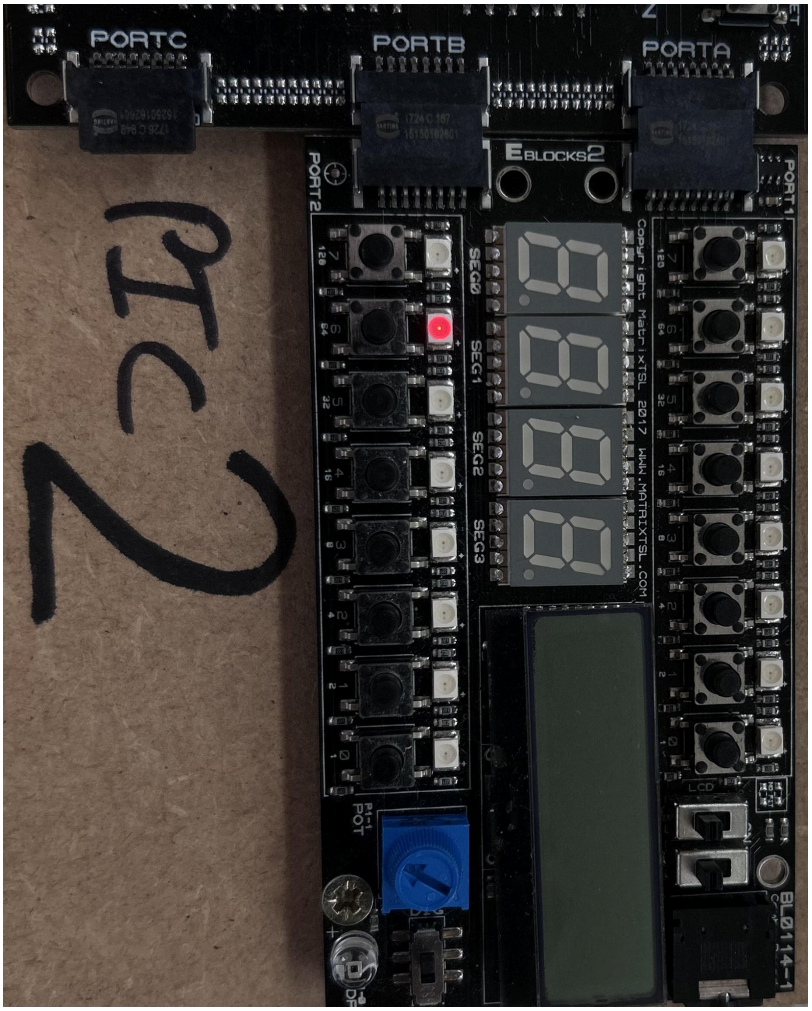

PORTA
PORTB
PORTC
EBLOCKS2
PORT1
PORT2
SEG0
SEG1
SEG2
SEG3
Copyright MatrixTSL 2017 www.matrixtsl.com
LCD
POT
DIG
BL0114-1
PIC

Interrupt:

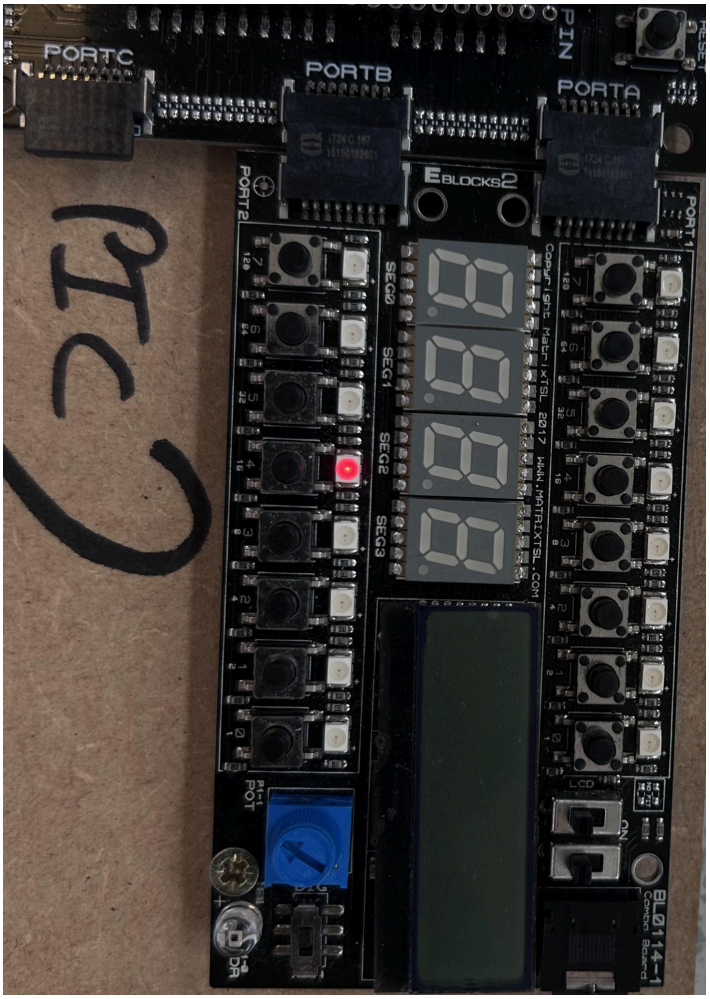


Vehicle Locking Pattern:


PORTC
PORTB
PORTA
EBLOCKS2
Copyright MatrixTSL 2017 WWW.MATRIXTSL.COM
SEG0
SEG1
SEG2
SEG3
PORT2
POT
BL0114-1
PIC 2

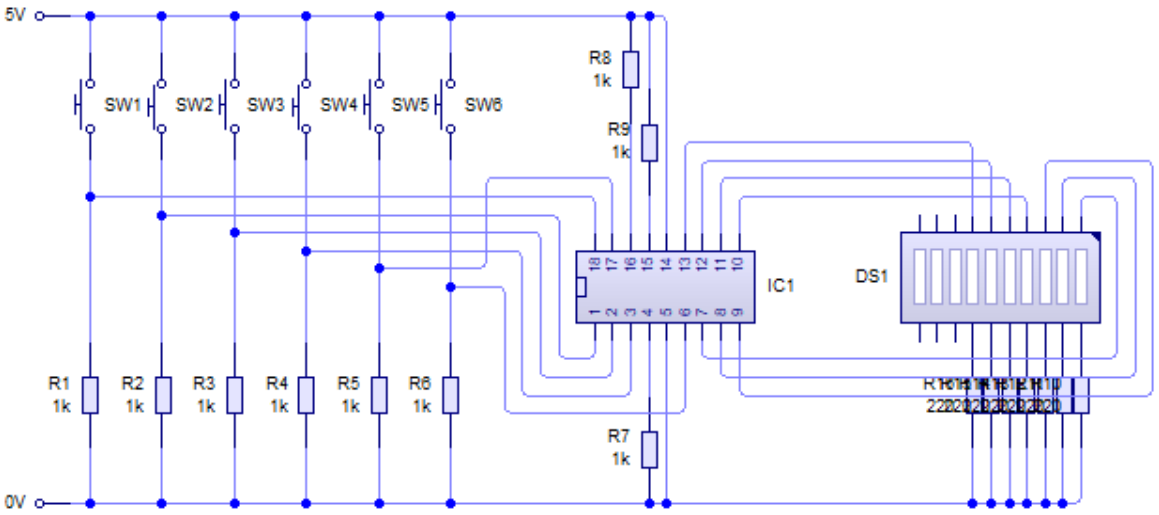


Emergency Services Call:

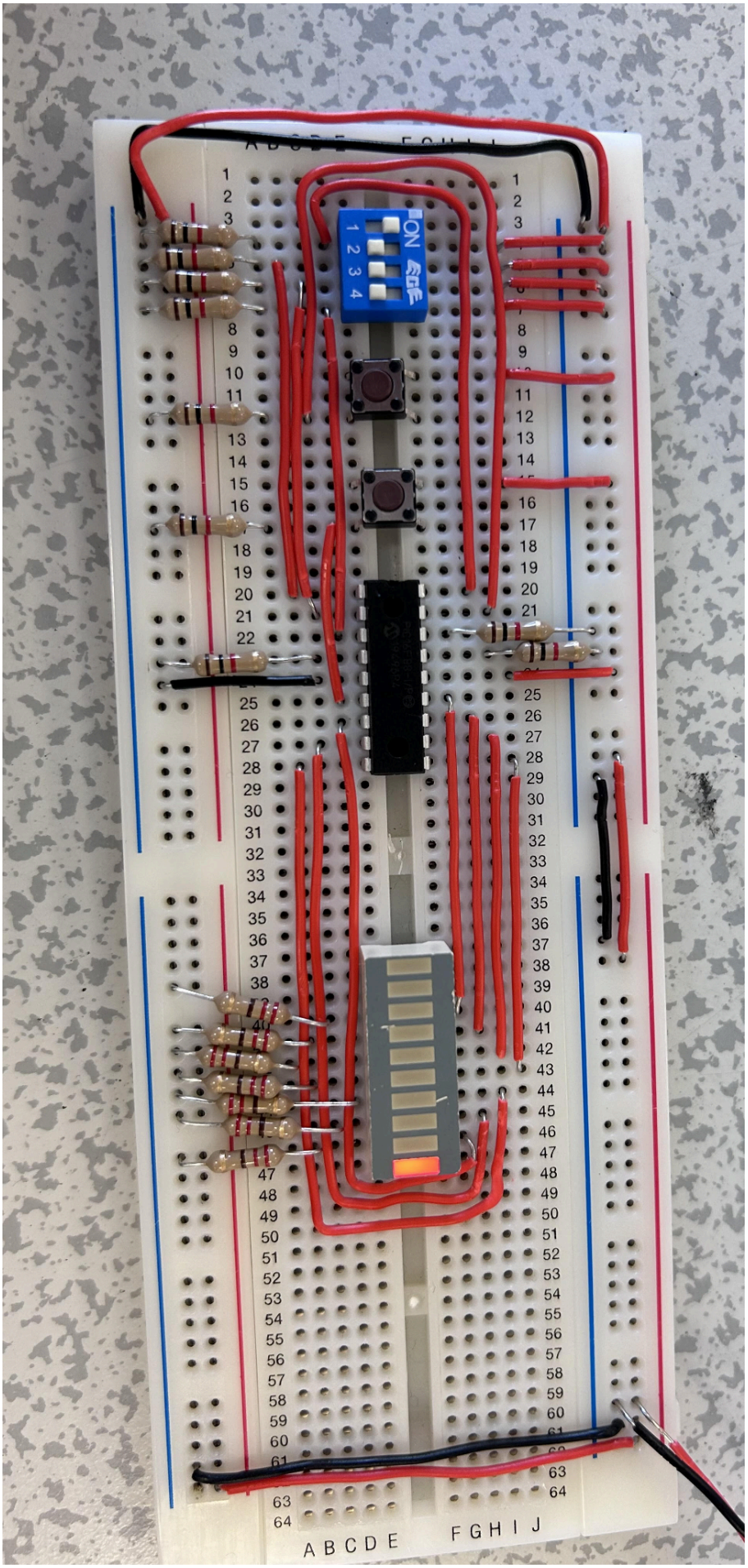


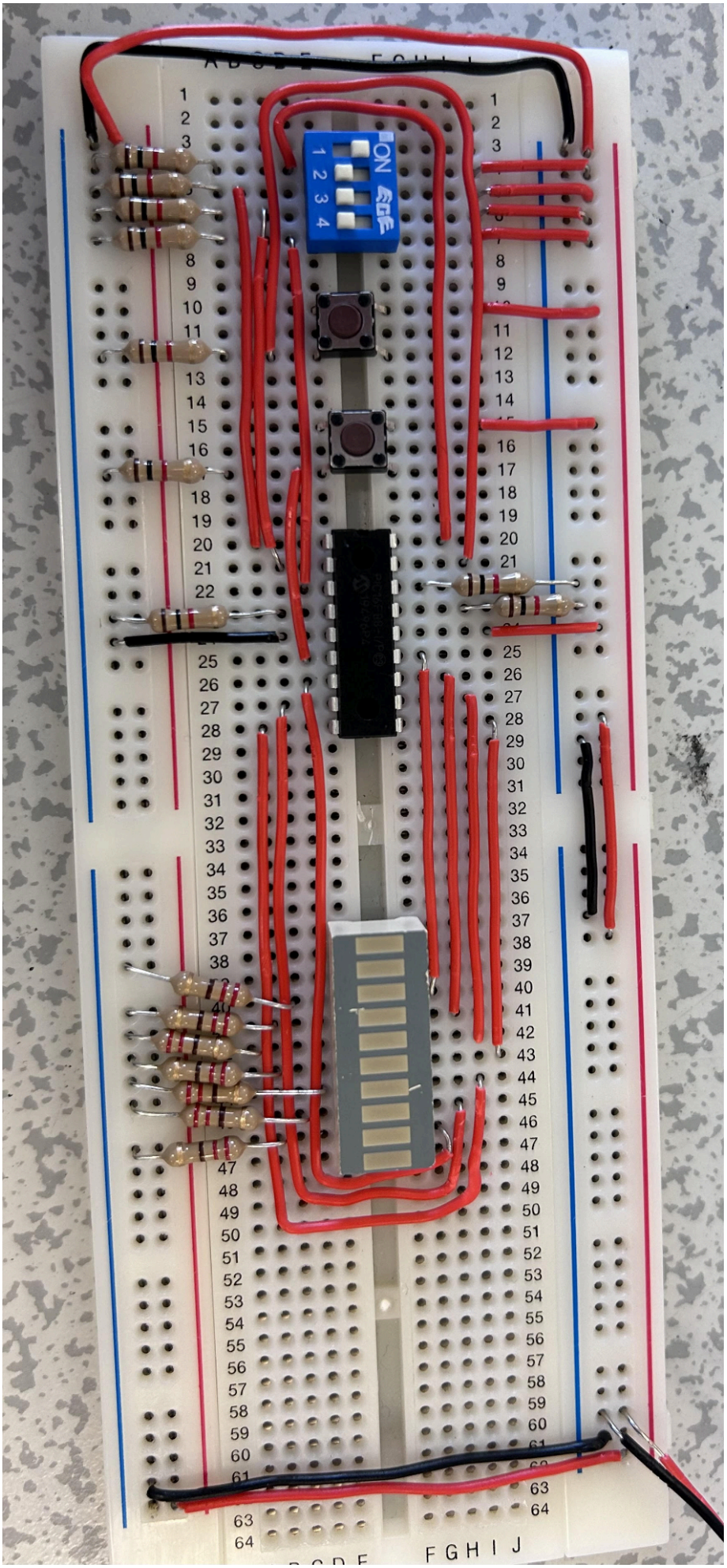
7) Photo of Circuit

Photo of circuit using circuit wizard program:

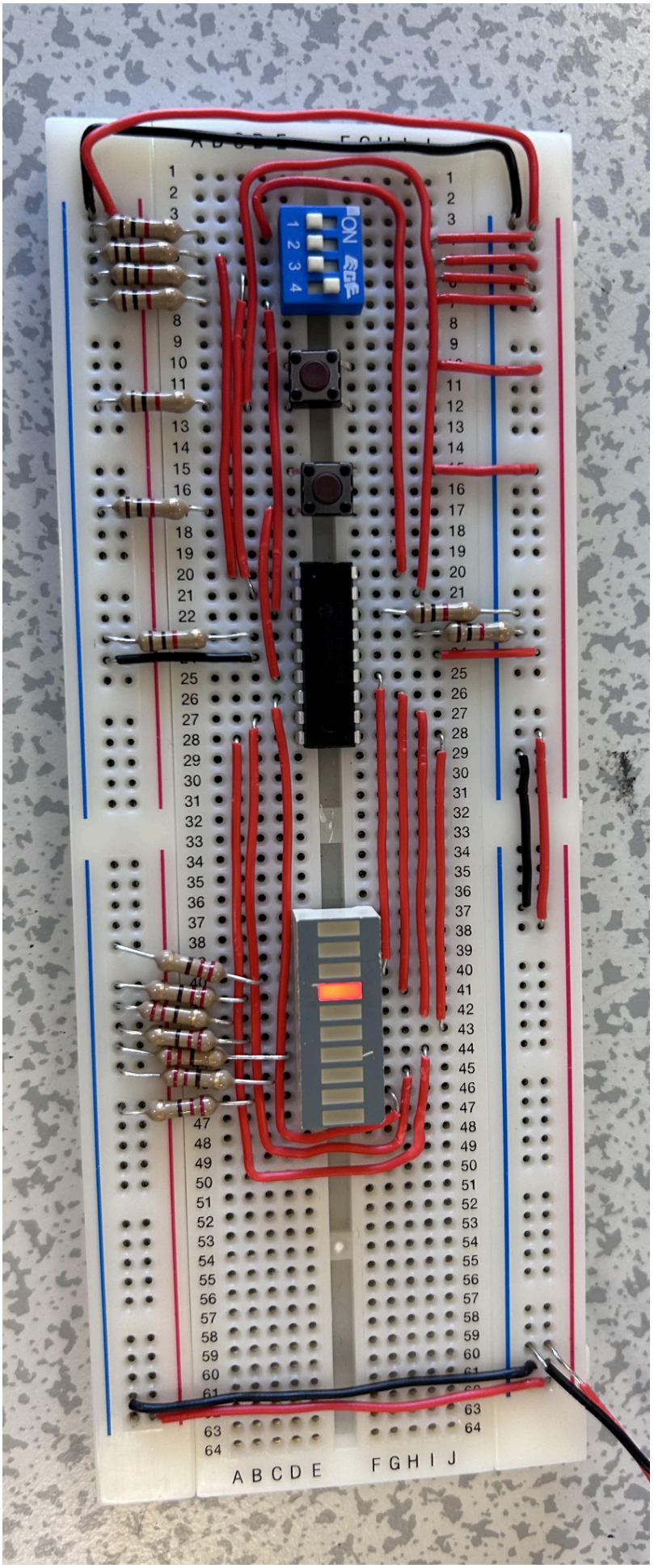


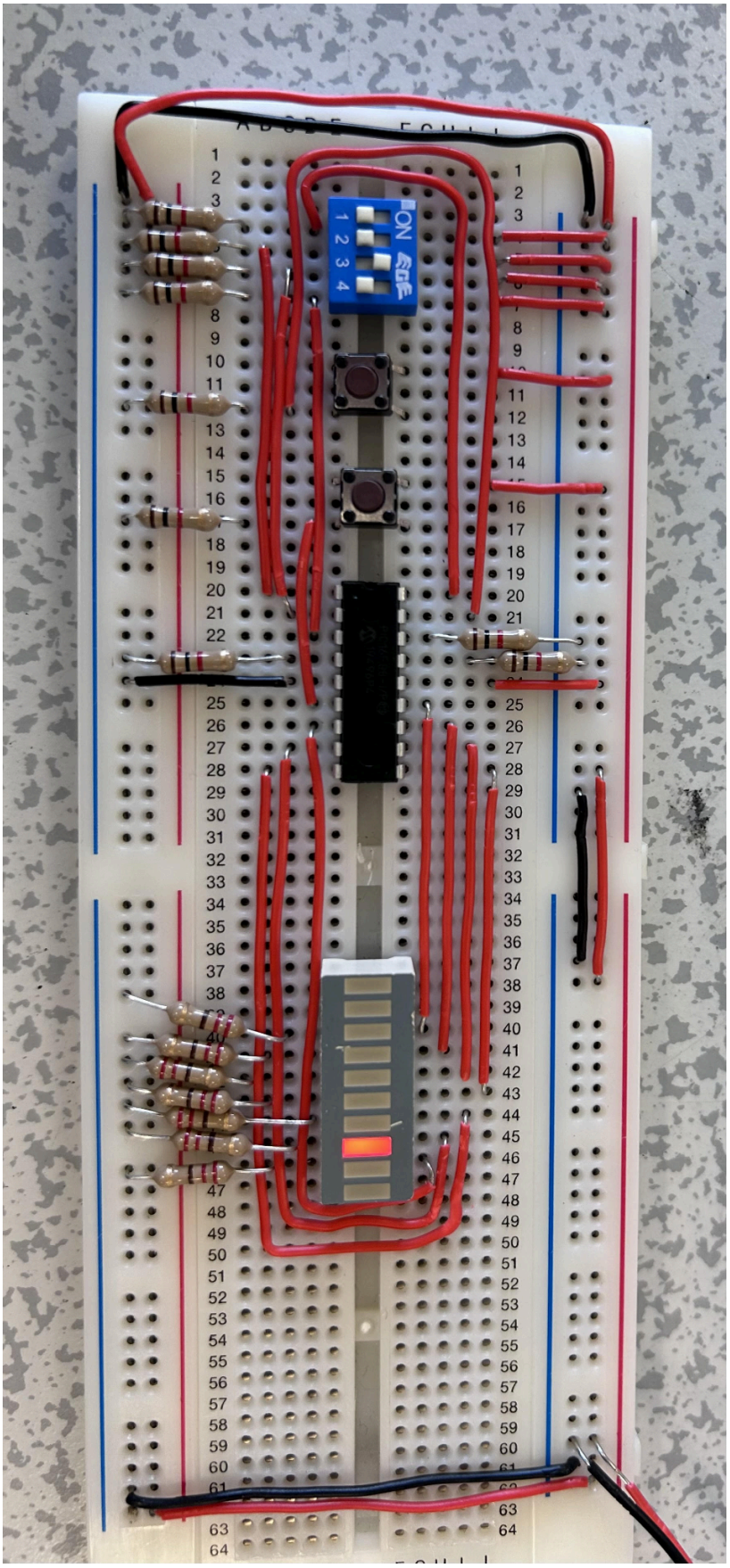
Impact / Door sensors and alarm


ON
1 2 3 4
A B C D E
F G H I J

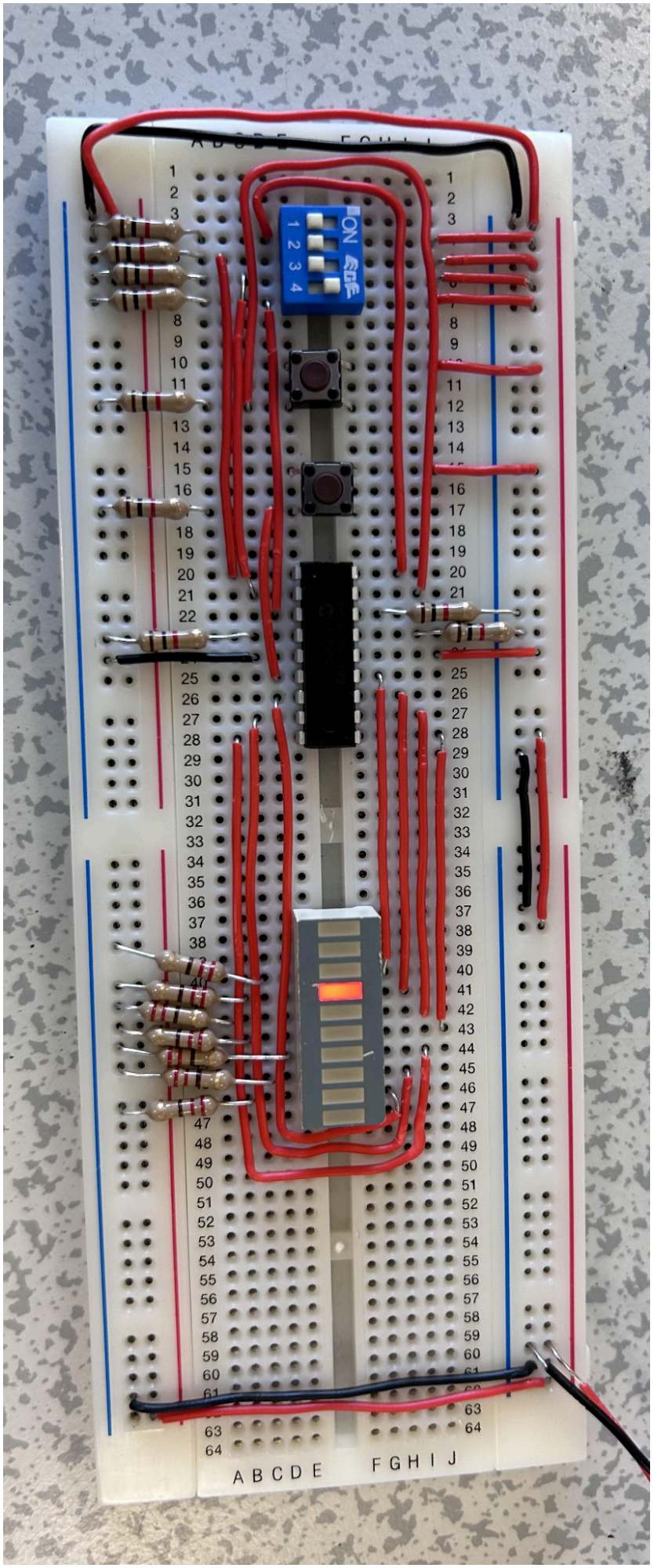

ON
1 2 3 4
F G H I J

Emergency services call

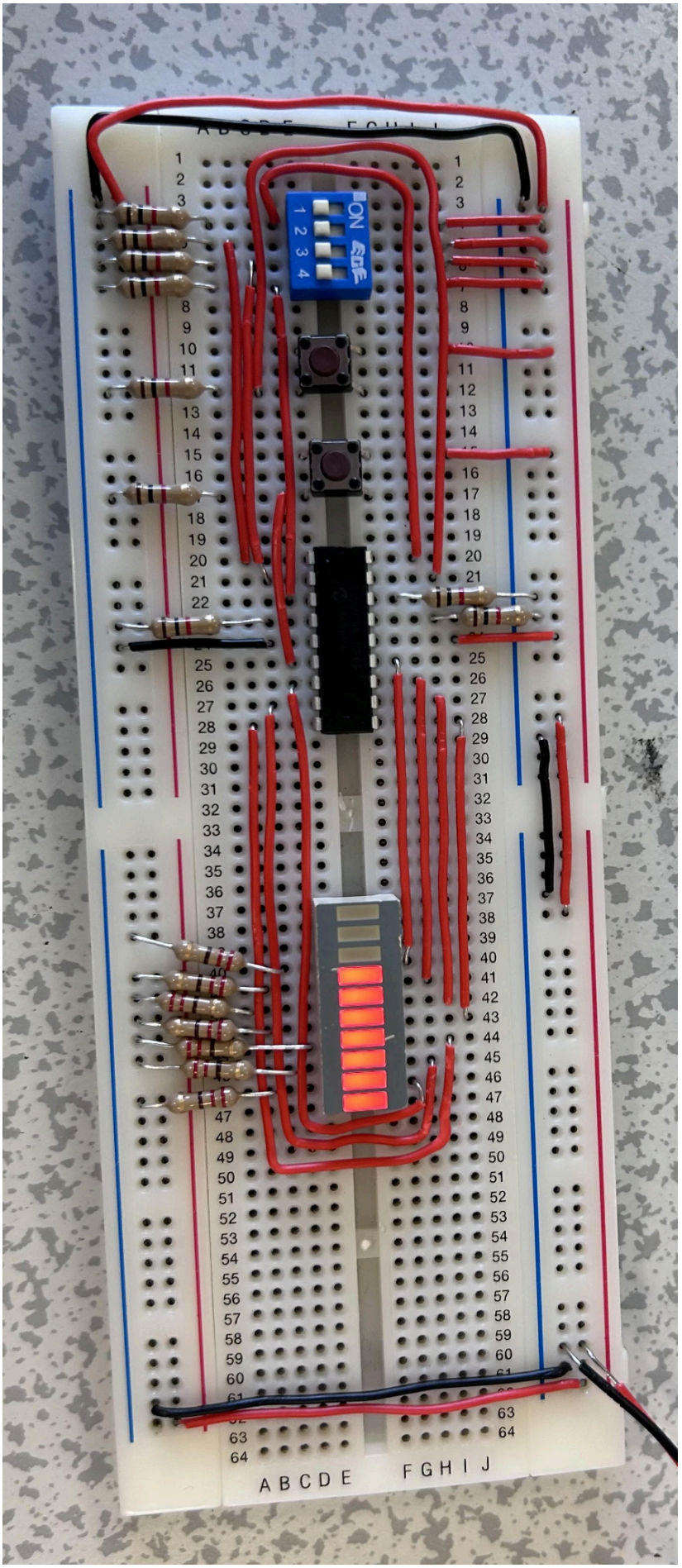

ON
1 2 3 4
A B C D E
F G H I J



Locking vehicle


ON
1 2 3 4
A B C D E
F G H I J

Interrupt


ON
1 2 3 4
A B C D E
F G H I J

8) Evaluation

In the end, my program achieved all of the goals which I originally set out to achieve.
There were a few changes made to the structure of the PIC system, but all of the baseline features worked as intended.

One of the features which I changed was to make the alarm use different times. At first, the alarm was continuous and did not end unless the reset button was pressed. However, due to this not being allowed, this was changed. Alarms are not allowed to be continuous and must have an interval of silence, this is why I changed the alarm to be 30 seconds of continuous buzzing, then a seven second silence between each repeat of the alarm.

Another feature of my PIC project which was changed was the locking vehicle system. From the first press of the lock vehicle button, all LEDs were to be turned on. However, this would have looked too similar to the interrupt routine and was changed to save confusion. After this change, the PIC system matched the expectations and planning which I made.

Compared to the rest of the specification, all of my objectives were met. The sensors which were connected to PORTA,1 and PORTA,2 worked as intended, setting an alarm for a period of time and could be reset at any time with the press of a button.
The emergency services call also managed to work as intended, it started a call loading pattern, followed by a 100 second timer before ending the call and returning to the start of the program.
The locking of the vehicle feature also worked, triggering a pattern which was continued until the reset button was pressed.

In the future, I would make more improvements in order for my system to work as effectively as possible. One example of an improvement I would add is to add a way which the car system is able to detect if any airbags have been deployed, if this is the case, then it will display a pattern similar to the interrupt and also automatically trigger the emergency services call system.

A second improvement I would make is changing the frequency of the alarm, as changing the frequency also changes the tone of the sound it makes, this would be useful as higher frequencies make for better alarms. As I have kept all frequencies the same, there is no variation in the alarms which the system produces.

Overall, my project meets most, if not all of the requirements which I set in the specification at the beginning of this Multi-Functional Car System project.