ELECTRONIC SYSTEM ENGINEERING DEPT.,
MALAYSIA-JAPAN INTERNATIONAL INSTITUTE OF TECHNOLOGY (MJIIT)
UNIVERSITI TEKNOLOGI MALAYSIA

SMJE 3183 – MICROPROCESSOR & MICROCONTROLLER

SEMESTER 1 – 2024/2025

PROJECT ASSIGNMENT [15%]

THIS PROJECT ASSIGNMENT WILL ASSESS YOU ON THE FOLLOWING ITEMS:

No.	CLO	PLO (Keyword)	Project Weightage (%)	Taxo. & generic skills*	WP	EA	WK	Assess-ment methods
CLO2	Ability to analyze a given task, identify task requirement and formulate solution in a programming in assembly language or C (option)	PLO 2 (ANALYSIS) THPA	5%	C4 TH-3	1		3	T2, ASG PR, F
CLO3	Apply to use Integrated Development Environment (IDE) to program, simulate, analyze a microcontroller program and to perform interfacing with peripherals	PLO 5 (MODERN TOOLS) SCMT	5%	P3-P4	3		6	Lab PR
CLO4	Work collaboratively and effectively in given tasks under the development of microcontroller- based embedded system	PLO 10 (TEAM WORK) – Report, - Presentation	5%	A3,TW1, CTPS2				PR

Develop an embedded system with AVR ATmega32 8-bit micro-controller using Arduino ATmega 256 board. The circuit design must be able to receive signals from at least 2 **different types** of input and results in outputs responses on at least 2 different types of device (e.g. LED light & sound buzzer) on the GPIO terminals. The design must include external hardware interrupt. No designation of any specific project theme. Let your creativity and innovativeness decide.

Prepare the following on the day of project presentation:

1) **Project Demonstration**

   <u>Hardware:</u>
   - Implementation on Arduino ATmega 256

   <u>Software</u>
   - Simulation using Microchip Studio IDE via I/O ports
   - Or using Proteus (extra marks)
   -

2) **PPT slides** that contains discussions on your project
   (i) Project title
   (ii) Project description (overall system) – project
   (iii) Project motivation
   (iv) Flowchart of work
   (v) Project coding

3) **Project Report (hardcopy)**
   (i) Introduction
   (ii) Project Objective – tangible output
   (iii) Project goals
   (iv) Project description – overview of of the system
   (v) Methodology
      a. Description on implementation on hardware platform
      b. description of individual input-output devices
      c. description of GPIO ports
      d. project circuitry
      e. coding with comments
   (vi) Conclusion

Please include and bind together this instruction sheet together with the report.

UTM
UNIVERSITI TEKNOLOGI MALAYSIA
Malaysia-Japan International Institute of Technology (MJIIT)

SEMESTER 1, SESSION 2022/2023

SMJE 3183 -

Microprocessor & Microcontroller

PROJECT ASSIGNMENT REPORT:

DRIVER SIMULATOR

NAME		Matric No.	Signature
1. HENRY LIM KIM YUAN		A22MJ0065	[illegible]
2. SYED ABRAR RAFID		A22MJ0004	[illegible]
Section number	**1**		
Group number	**8**		

(I) INTRODUCTION

In recent years, the number of driving learners in Malaysia have increased. According to Malaysia Ministry of Transport, from 2019 to 2023, number of registered vehicles have increased from 31 million to 36 million. Unfortunately the number of road accidents have also increased. From 2021 to 2022, number of road accidents have increased from 370286 to 545588. If this worrying trend is not addressed immediately, these numbers will continue to rise, increasing the risk towards road and vehicle users safety.

As most road accidents are either caused by poor driving habits, weak defensive driving skills or easily panicked drivers, it is essential to inculcate good driving habits and instill defensive driving in all drivers as early as their learning phase. Unfortunately, most driving learners only have a maximum of 22 hours to learn their driving skills before taking their test. In this short amount of time, they have to cram plenty of driving knowledge and get used to everything in the car which could lead to improper mastery of key driving skills, resulting in dangerous drivers on the road.

Therefore, our group noticed an indirect demand for easier and safer learning tools that can build confidence in driving learners and facilitate their mastery of good driving skills better. As of recent, driving simulators have proved to be an effective tool in streamlining a driving learner's progress in picking up driving skills. The simulator provides learners with a safe and low stress virtual environment to practice and make mistakes without sustaining physical harm to themselves. This enables drivers to learn faster from their mistakes without the unnecessary stress from the yelling and cussing of some driving instructors with poor ethics. As such, our group aims to develop an AVR embedded system driver simulator using Arduino Mega, KY-023 joystick module, KY-004 pushbutton module, buzzer and LEDs.

(II) PROJECT OBJECTIVE – TANGIBLE OUTPUT

This project aims to achieve several tangible outcomes, which can be physically observed from our completed AVR embedded system:

1. Develop a joystick embedded system that takes in input from an arduino joystick module and displays the output by lighting up LEDs corresponding to the direction of the joystick movement.
2. Design an external interrupt system that is activated via pushbutton and displays the interrupt service routine through toggling a LED on/off.
3. Build a pushbutton module embedded system that receives input from a arduino pushbutton module and displays the output by sounding an active buzzer.

(III) PROJECT GOALS

In line with our motivation of developing a driving simulator embedded system and our project objectives, our AVR embedded system also aims to achieve the following project goals:

1. Develop a driver simulator to provide driving learners with a safe environment to begin learning and adapting to the basics of driving a vehicle.
2. Design a very simple and beginner friendly control panel consisting of just a joystick as vehicle steering device and 2 buttons (one for activating emergency brake and one for horn) to streamline the learning process of beginner driving learners.
3. Build an emergency brake button that driving learners can activate anytime they lose control of their vehicle during simulation.

(IV) PROJECT DESCRIPTION – OVERVIEW OF OF THE SYSTEM

This AVR embedded system is comprised of 3 key systems: The vehicle steering system (modelled by KY-023 joystick module and 4 LEDs) , the emergency braking system (modelled by push button and 1 LED) and the horn system (modelled by KY-004 pushbutton module and an active buzzer).

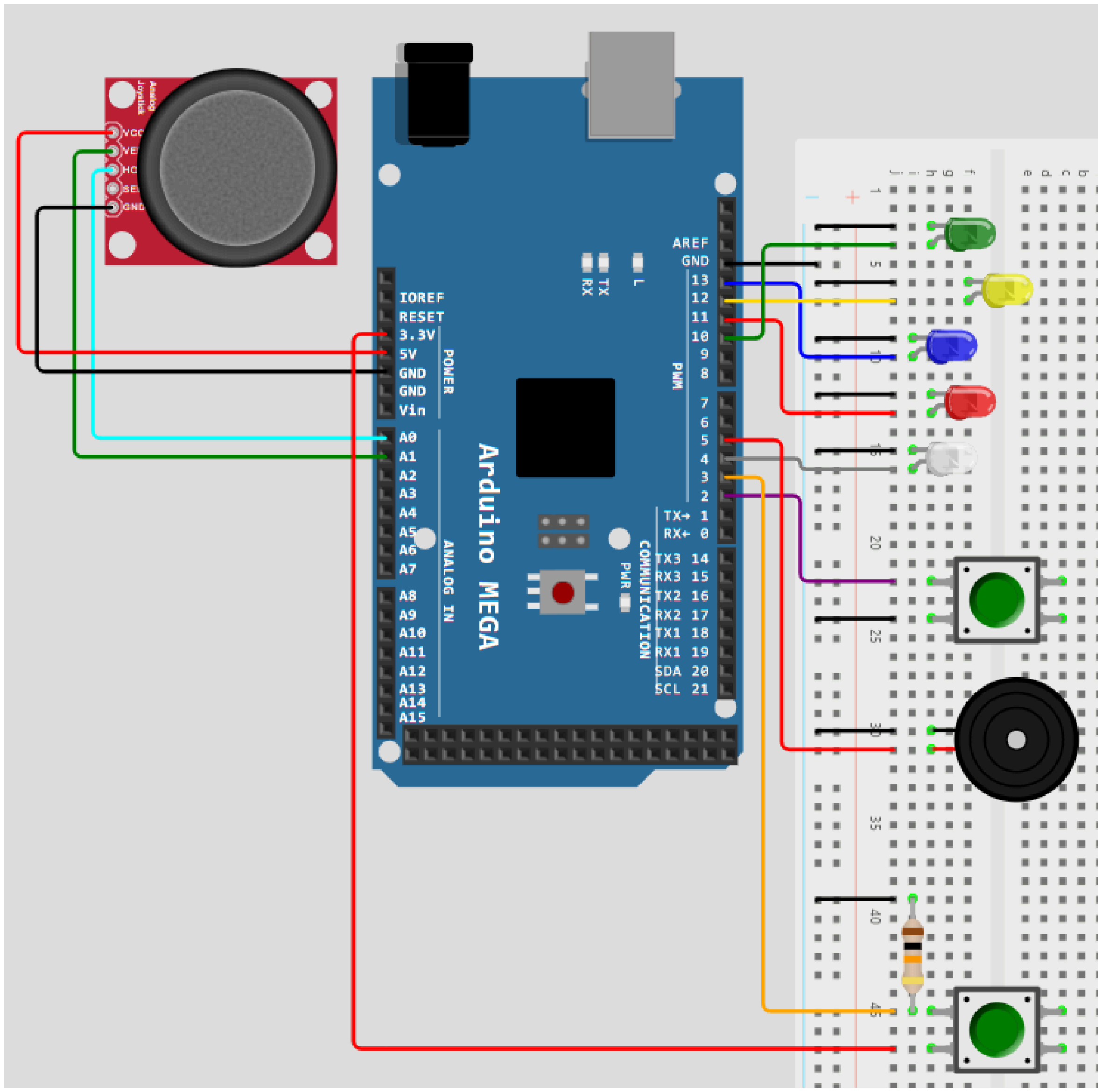
With regards to the vehicle steering system, the input sensor used is an analog sensor (KY-023 joystick module). This joystick module is meant to represent a steering wheel which controls the moving direction of a vehicle. Since the joystick is an analog sensor, thus any information sent to the ATMEGA 2560 microcontroller is first converted from analog input values to digital input values using ADC converter at the analog input ports on the Arduino Mega. On the other hand, the type of output device is a LED which can be lit up or turn off using just digital values. The 4 LEDs will each represent a direction (either forward, backward, left or right) to indicate the direction that the vehicle is moving. The AVR assembly firmware written on ATMEGA 2560 microcontroller then connects the input to the output by processing the input values received from joystick module to determine which corresponding LED should be lit up to match the direction the joystick has been moved to.

As for the emergency braking system, the input sensor is a simple push button. The push button immediately brakes the car. When the pushbutton is pressed again, it releases the brakes. The output will consist of 1 output device: LED. The LED indicates that the emergency brake has been activated. Since this system is for emergency purposes, it needs to be able to respond immediately upon activation irrespective of what the driver was previously doing. Thus, an external interrupt was implemented for this system where the pushbutton is connected to an external interrupt port, and the interrupt service routine (ISR) is the illumination of the LED. When the driving learner has lost control of his vehicle and wishes to stop the vehicle, he can press the push button to trigger the external interrupt which is detected by the external interrupt pin connected to the push button. The AVR assembly firmware written on ATMEGA 2560 microcontroller then momentarily halts the vehicle steering operation and immediately executes the ISR before returning to the vehicle steering operation.

For the horn system, the input sensor is a KY-004 pushbutton module. The pushbutton module represents a horn button which sounds the horn whenever the horn is pressed. When the push button is released, the horn will stop making sound. The output device is an active buzzer which represents the horn. The horn makes a sound whenever horn button is pressed to indicate that the horn button is pressed. The AVR assembly firmware written on ATMEGA 2560 microcontroller connects the input to the output by evaluating whether the input values received is HIGH (1) signal or LOW (0) signal, if the input is HIGH, a HIGH signal is also sent to the active buzzer causing the buzzer to make a sound. If the input is LOW, a LOW signal is also sent to the active buzzer causing the buzzer to stop making a sound.

(V) METHODOLOGY

A. DESCRIPTION ON IMPLEMENTATION ON HARDWARE PLATFORM



The image above is the physical implementation of our AVR embedded system on the hardware platform Arduino Mega prepared using Wokwi online simulator. In summary, the description of implementation will be divided into 3 parts: the vehicle steering system, emergency braking system and horn system. Since the KY-004 pushbutton module is unavailable in Wokwi, ot is modelled using a push button and a 10kΩ resistor.The table below summarises the pin connection of each component in each system to the Arduino Mega:

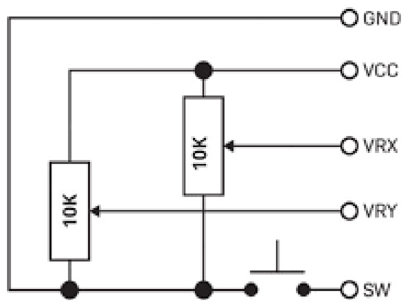
SYSTEM	COMPONENT	COMPONENT PIN	ARDUINO PORT	ATMEGA 2560 PIN
VEHICLE STEERING	KY-023 JOYSTICK MODULE	VCC	5V	NOT RELATED
		GND	GND	NOT RELATED
		VX	A0	PF0
		VY	A1	PF1
	FORWARD LED	ANODE	D12	PB6
		CATHODE	GND	NOT RELATED
	BACKWARD LED	ANODE	D13	PB7
		CATHODE	GND	NOT RELATED
	LEFT LED	ANODE	D10	PB4
		CATHODE	GND	NOT RELATED
	RIGHT LED	ANODE	D11	PB5
		CATHODE	GND	NOT RELATED

SYSTEM	COMPONENT	COMPONENT PIN	ARDUINO PORT	ATMEGA 2560 PIN
EMERGENCY BRAKE	PUSH BUTTON	PIN 1	D2 (INT4)	PE4
		PIN 2	GND	NOT RELATED
	INDICATOR LED	ANODE	D4	PG5
		CATHODE	GND	NOT RELATED
HORN	KY-004 PUSH BUTTON MODULE	SIGNAL (S)	D3	PE5
		+VIN	GND	NOT RELATED
		GND	5V	NOT RELATED
	ACTIVE BUZZER	ANODE	D5	PE3
		CATHODE	GND	NOT RELATED

B. DESCRIPTION OF INDIVIDUAL INPUT-OUTPUT DEVICES

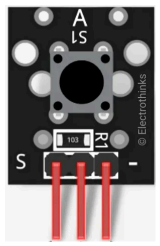
In this AVR embedded systems project, 2 types of input devices were used (KY-023 Joystick module & KY-004 Push Button module) and 2 types of output devices (LED and Active Buzzer) were used. For the external interrupt system (emergency brake), 1 type of input device (push button) and 1 type of output device (LED) was used. This section discusses these devices individually.

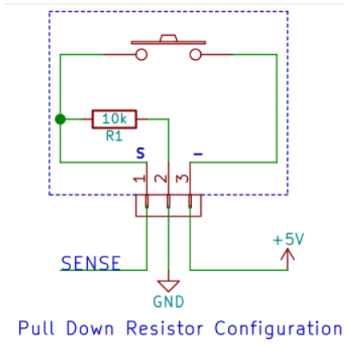
KY-023 Joystick Module:



As seen from the schematic diagram above, the joystick module comprise of 2 potentiometers: one for the x-axis (left & right), one for the y-axis (forward & backward). The VRX and VRY pins will each output a fraction of the VCC input voltage based on position of wiper on their respective resistive tracks. In our project, we will connect VRX and VRY pins to analog pin A0 and A1 because the change of output voltage from potentiometer is analog. The ADC in pin A0 and A1 will convert these analog values into digital values which can be processed by ATMEGA 2560 microcontroller. SW pin will not be used in our project as we are not using the push button of the joystick. In our project, the joystick is used as a steering device to control the direction of the vehicle’s motion.

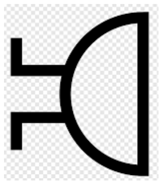
KY-004 Push Button Module:





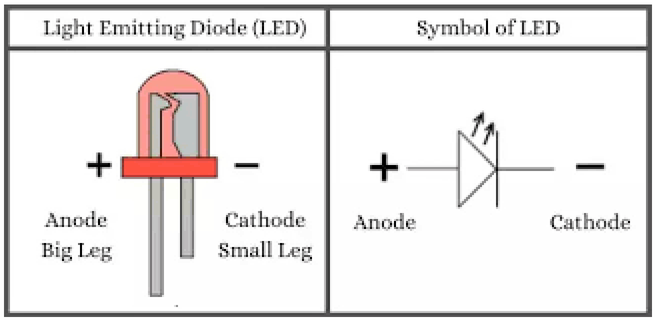
The KY-004 push button module consists of a 10k internal resistor and a pushbutton. In our project, we modified the pushbutton module into a pull down resistor setup. Hence, the cathode (negative) pin is connected to the +5V pin while the +Vin is connected to ground to complete the pull down resistor modification. In the pull down resistor configuration, circuit is initially open with no current flowing to sense (S) pin and +Vin pin connected to ground. S pin which is connected to pin D5 of Arduino Mega inputs a LOW (0) signal. When the pushbutton is pressed, the pull down resistor circuit is closed and current flows from cathode pin to +Vin pin which is connected to ground and S pin which is connected to pin D3 of the Arduino Mega. Now, the S pin sends a HIGH (1) input signal to pin D3 of Arduino Mega. Conversely, when the pushbutton is released, the pull down resistor circuit is open and current flows from cathode pin to the pushbutton only. Now, the S pin sends a LOW (0) input signal to pin D3 of Arduino Mega. In our project, the KY-004 push button module forms the horn button of our horn system.

Active Buzzer:



An active buzzer has a built-in oscillator, making the requirements to sound the buzzer as simple as supplying a DC voltage to the active buzzer. Once the DC voltage is supplied, the buzzer makes a sound. They are simpler to use than their passive counterparts and hence the active buzzer was chosen due to its simplicity. The active buzzer has its polarity which is anode (positive) and cathode (negative). In our project, the buzzer acts as the horn in the horn system. Its anode is connected to pin D5 and cathode is connected to ground.

Light Emitting Diode (LED)



The LED is a simple diode that emits light when current pass through it. It is often used as an indicator for presence of current flow. It has its polarity as well: anode (positive) and cathode (negative). In our project, the LED is used in both the vehicle steering system and emergency braking system. In the vehicle steering system, the LED will represent each direction (forward, backward, left or right) to indicate which direction the vehicle is moving to match the direction the joystick has been moved to. In emergency braking system, the role of the LED is to indicate that the emergency brake has been activated.

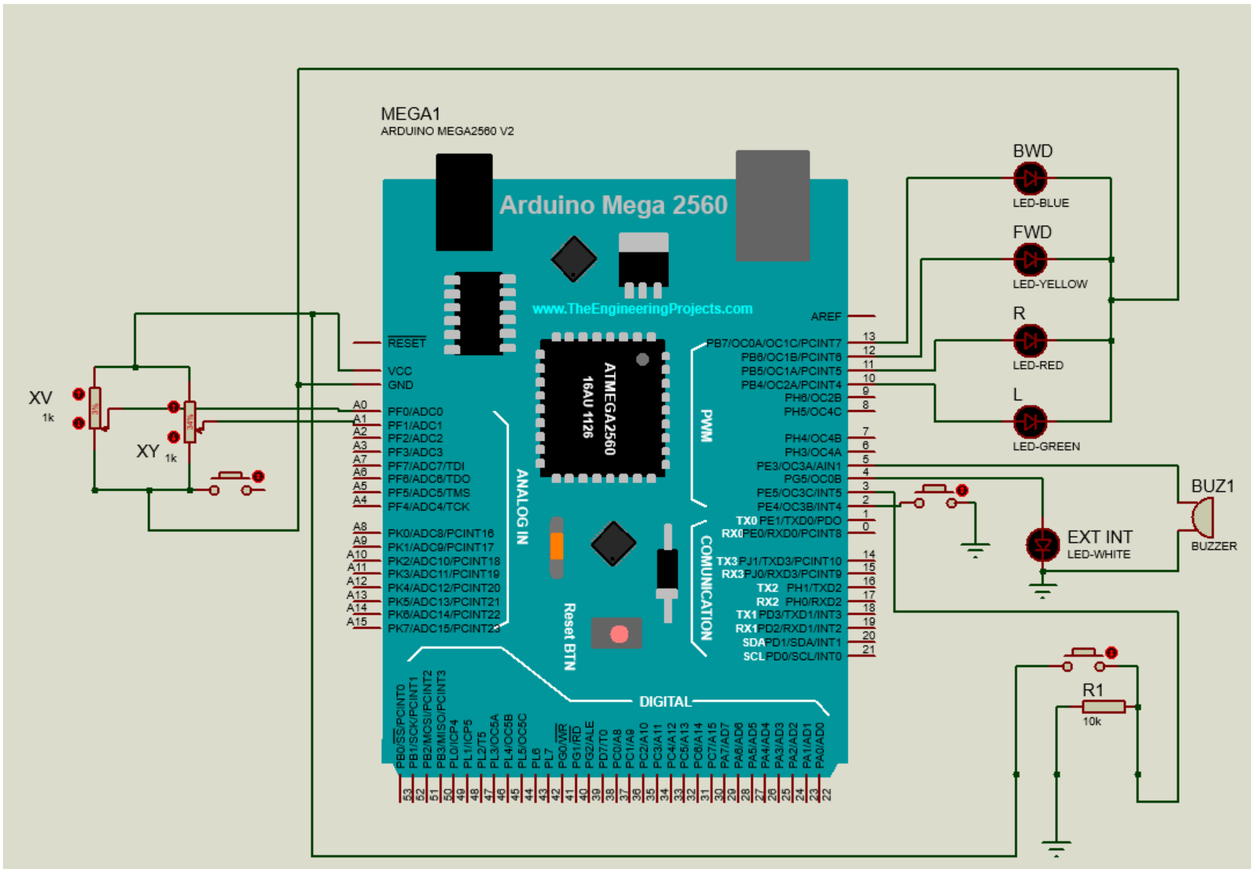
C. DESCRIPTION OF GPIO PORTS

A description of all the GPIO Ports used will be discussed in this section.

GPIO Port (Arduino Mega)	ATMEGA 2560 Port	Description / Function
D2 (INT4)	PE4	Act as external interrupt pin. Within the Arduino Mega board, Port E bit 4 is internally connected to a pull up resistor forming an active low circuit. When push button is pressed, the active low circuit is completed and temporarily forms a low signal at Port E bit 4, triggering the external interrupt.
D4	PG5	When external interrupt is triggered, this output port toggles its output between HIGH/LOW to the buzzer and LED in the external interrupt system.
D3	PE5	Takes input from pull down resistor in KY-004 pushbutton module. When pushbutton is pressed, the pushbutton circuit is closed and pushbutton module sends a HIGH signal to pin D3 while pushbutton is still pressed.
D5	PE3	Outputs HIGH signal to active buzzer when pushbutton is pressed.
D10	PB4	Outputs HIGH signal to LED connected when joystick is pushed to the left.
D11	PB5	Outputs HIGH signal to LED connected when joystick is pushed to the right.
D12	PB6	Outputs HIGH signal to LED connected when joystick is pushed forward.
D13	PB7	Outputs HIGH signal to LED connected when joystick is pushed backward.
A0	PF0	Reads analog signal (from 0V to 5V) from VRX potentiometer pin in joystick module and converts it to digital value of (0 - 1023) using ADC converter.
A1	PF1	Reads analog signal (from 0V to 5V) from VRY potentiometer pin in joystick module and converts it to digital value of (0 - 1023) using ADC converter.

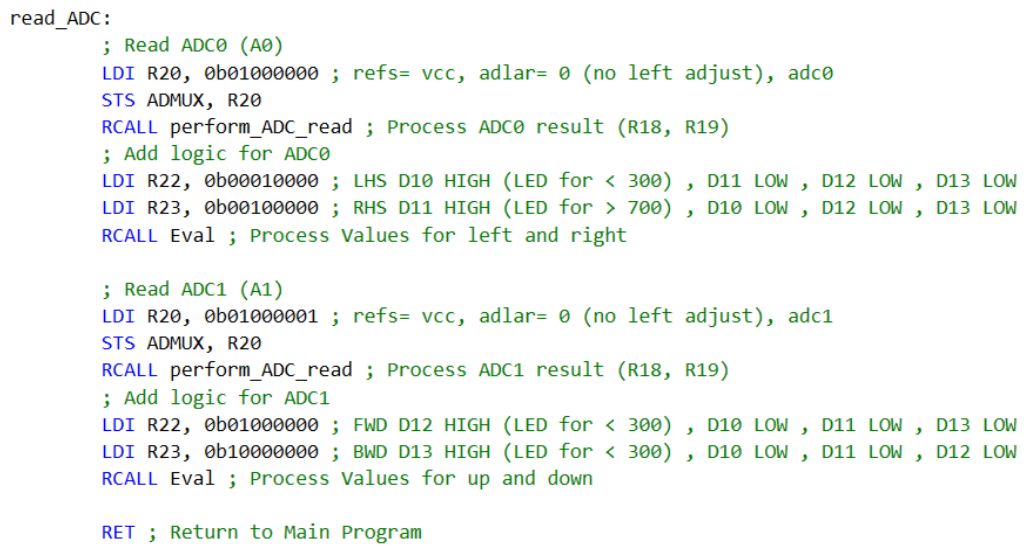
D. PROJECT CIRCUITRY

For the project circuitry, a schematic diagram was prepared using Proteus software. Since Proteus software does not have library for KY-023 joystick module and KY-004 pushbutton module, so KY-023 is modelled using 2 potentiometers and a push button while KY-004 is modelled using a push button and a 10kΩ resistor.

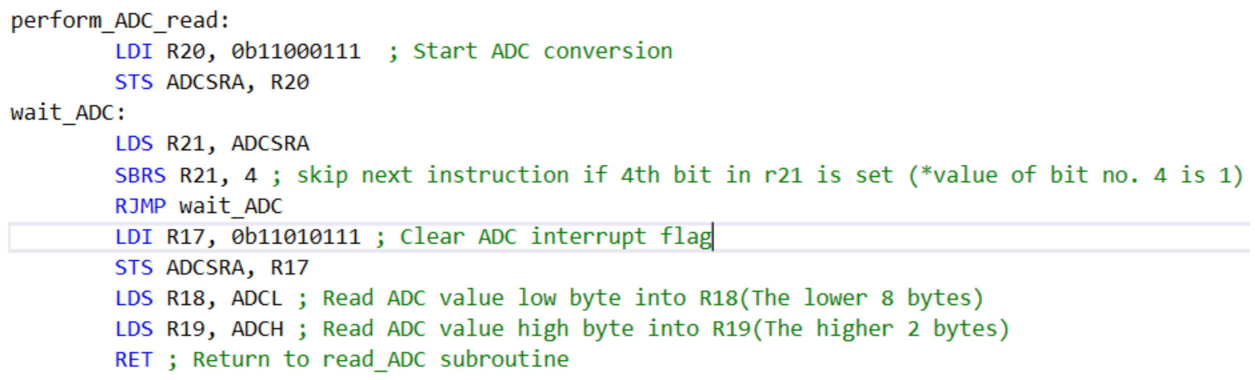


E. CODING WITH COMMENTS

```
        .INCLUDE "m2560def.inc" ; Include the mapping library for ATMEGA2560 and Arduino Mega
        .CSEG                   ;Indication to assembler that this is the start of a code segment
        .ORG 0x0000             ;Location for reset
        JMP Main                ;Jump to location of Main Program
        .ORG 0x000A             ;Location for external interrupt 4
        JMP externalISR4        ;Jump to location of ISR

Main:
        ; Configure External Interrupt INT4
        Ldi R20,HIGH(RAMEND);Set up the stack
        Out SPH,R20             ;Set up the stack
        Ldi R20,LOW(RAMEND) ;Set up the stack
        Out SPL,R20             ;Set up the stack
        LDI R20, 0b00000010 ; Enable INT4(Pin D2) on falling edge (ISC01=1, ISC00=0)
        STS EICRB, R20          ; For INT4, Use Enable Interrupt Control Register B
        LDI R20, 0b00010000 ; Enable external interrupt INT4 (Bit 5 of Enable Interrupt Mask Register)
        OUT EIMSK, R20
        SEI ; Enable global interrupts (I flag in SREG is set to 1)
        SBI PORTE,4             ;Activated pull-up internal resistor for pin D2 as ext interrupt

        CALL init ; Call init subroutine
loop:   IN R24, PINE ; take in reading from pushbutton to D3
        CALL HORN ; Call HORN subroutine
        CALL read_ADC ; Call read_ADC subroutine
        RJMP loop ; Continuously read ADC values from Joystick module
end:    RJMP end ; End of code, stay here

;From here onwards we are just defining subroutines

init:   ; Set pinModes
        CBI DDRF, 0 ; A0 (input)
        CBI DDRF, 1 ; A1 (input)
        SBI DDRE, 3 ; set D5 as o/p
        CBI DDRE, 5 ; set D3 as i/p
        LDI R20, 0b11110000 ; set D10 (PB4, LHS), D11 (PB5, RHS), D12 (PB6, FWD) and D13 (PB7, BWD)as outputs
        OUT DDRB, R20
        ; Set Up ADMUX(ADC Multiplexer Selection Reg.) and ADCSRA(ADC Control & Status Reg. A)
        LDI R20, 0b01000000 ; bit7&6: voltage ref select = vcc (refs1=0, refs0=1), bit5: adlar= 0 (right justified, ADCH=2bit, ADCL=8bit),
                            ; bit3-0: analog channel select = adc0 (mux3=0, mux2=0, mux1=0, mux0=0)
        STS ADMUX, R20
        LDI R20, 0b10000111 ; bit7: enable adc(1), bit6: don't start ADC convert(0), bit5: don't auto trigger(0), bit4: conver. not done (0),
                            ; bit3: do not activate ADC conver. interrupt(0), bit2-0: choose slowest conv speed but highest accuracy, div128 (111)
        STS ADCSRA, R20
        RET ; Return to Main program
```



```
read_ADC:
        ; Read ADC0 (A0)
        LDI R20, 0b01000000 ; refs= vcc, adlar= 0 (no left adjust), adc0
        STS ADMUX, R20
        RCALL perform_ADC_read ; Process ADC0 result (R18, R19)
        ; Add logic for ADC0
        LDI R22, 0b00010000 ; LHS D10 HIGH (LED for < 300) , D11 LOW , D12 LOW , D13 LOW
        LDI R23, 0b00100000 ; RHS D11 HIGH (LED for > 700) , D10 LOW , D12 LOW , D13 LOW
        RCALL Eval ; Process Values for left and right

        ; Read ADC1 (A1)
        LDI R20, 0b01000001 ; refs= vcc, adlar= 0 (no left adjust), adc1
        STS ADMUX, R20
        RCALL perform_ADC_read ; Process ADC1 result (R18, R19)
        ; Add logic for ADC1
        LDI R22, 0b01000000 ; FWD D12 HIGH (LED for < 300) , D10 LOW , D11 LOW , D13 LOW
        LDI R23, 0b10000000 ; BWD D13 HIGH (LED for < 300) , D10 LOW , D11 LOW , D12 LOW
        RCALL Eval ; Process Values for up and down

        RET ; Return to Main Program
```



```
perform_ADC_read:
        LDI R20, 0b11000111  ; Start ADC conversion
        STS ADCSRA, R20
wait_ADC:
        LDS R21, ADCSRA
        SBRS R21, 4 ; skip next instruction if 4th bit in r21 is set (*value of bit no. 4 is 1)
        RJMP wait_ADC
        LDI R17, 0b11010111 ; Clear ADC interrupt flag
        STS ADCSRA, R17
        LDS R18, ADCL ; Read ADC value low byte into R18(The lower 8 bytes)
        LDS R19, ADCH ; Read ADC value high byte into R19(The higher 2 bytes)
        RET ; Return to read_ADC subroutine
```

```
Eval:
        ; Now we have the full 10 bit ADC value in R18 (low byte) and R19 (high byte)
        ; The full ADC value is (R19 bit 1 and bit 2) + R18

        ; Check if ADC value < 300
        LDI R20, 0b01
        ; Compare high byte first
        CP R19, R20 ; Compare ADC value high byte with 1st 2 MSB of 300
        BREQ check_low ; If high byte is equal, check low byte
        BRLO LU ; If overall ADC value less than 300, go to Left(D10) or Up(D12)

        ; Check if ADC value > 700
        LDI R20, 0b10 ; Load R20 with 1st 2 MSB of 700
        CP R19, R20 ; Compare ADC value high byte with with 1st 2 MSB of 700
        BREQ cont_check ; Continue checking LSB of ADC value
cont_check: LDI R20, 0b10111100 ; Load R20 with the remaining 8 LSB for 700
            CP R18, R20 ; Compare ADC value low byte with remaining 8 LSB for 700
            BRSH RD ; If greater than or equal to 700, go to Right(D11) or Down(D13)
            BRLO idle ; If Lower than 700, go to Idle

check_low:
        ; If ADC value high byte is equal to first 2 MSB of 300, check the low byte
        LDI R20, 0b00101100 ; Load R20 with the remaining 8 LSB for 300
        CP R18, R20 ; Compare ADC value low byte with remaining 8 LSB for 300
        BRLO LU ; If less than 300, go to Left(D10) or Up(D12)

idle:
        ; If ADC value is between 300 and 700, turn off both LEDs
        CLR R20 ; Turn off all LEDs
        STS PORTB, R20 ; Turn off all LEDs
        RJMP exit ; Go to Exit

LU:
        OUT PORTB, R22 ; Either Left or Forward LED will light up while others will turn off
        RJMP exit ; Go to Exit

RD:
        OUT PORTB, R23 ; Either Right or Backward LED will light up while others will turn off
        RJMP exit ; Go to Exit

exit:       RET ; Exit EVAL subroutine back to read ADC soubroutine
```

```
HORN:
            SBRC R24, 5 ; skip next step if D3 = 0
            RJMP Over
            CBI PORTE, 3 ; digitalWrite(5, LOW)
            RJMP Here
Over:       SBI PORTE, 3 ; digitalWrite(5, HIGH)
            RJMP Here
Here:       RET ; Return to Main Program

/* ---------- Interrupt Service Routine ------------ */
externalISR4:
            SBI DDRG, 5 ; D4 (output) , LED for interrupt
            IN R17, PORTG ; Read PORTG
            LDI R16, 0b00100000 ; Set bit 5 with 1, if bit 5
            EOR R17, R16 ; EOR Port G bit 5 with bit 5 of R16: If PORTG bit5 is 1, 0(LOW) will be saved into R17 bit5,
                         ; If PORTG bit5 is 0, 1(HIGH) will be saved into R17 bit5
            OUT PORTG, R17  ; Output value of bit 5 in R17 to PortG bit5
            Reti ; Exit ISR back to main program
```

(VI) CONCLUSION

In conclusion, to fulfill the demands of producing drivers with defensive driving skills and good driving habits, a safe and stress free learning tool for drivers (in the form of driving simulator) is required. A joystick embedded system that takes in input from an arduino joystick module and displays the output by lighting up LEDs corresponding to the direction of the joystick movement was successfully developed. Thus, a very simple and beginner friendly control panel to streamline the learning process of beginner driving learners was successfully designed. Moreover, an external interrupt system that is activated via pushbutton and displays the interrupt service routine through toggling an active buzzer on/off was able to be constructed. Hence, an emergency break plus horn button that driving learners can activate anytime they lose control of their vehicle during simulation is built. Lastly, both AVR embedded systems above were enhanced by combining the joystick embedded system with the external interrupt system using Arduino Mega (ATMEGA 2560 microcontroller) so that both systems can operate simultaneously. The end result is a well engineered driver simulator that provides driving learners with a safe environment to begin learning and adapting to the basics of driving a vehicle. We hope that this innovation could benefit more driving learners to overcome their fears and anxiety of driving and make them more confident and competent safe drivers.