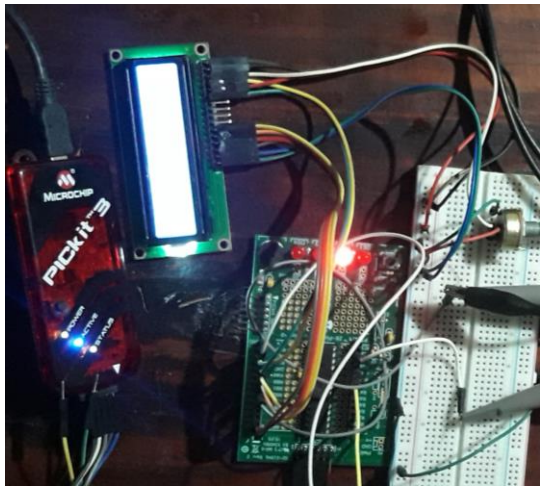
PIC programming

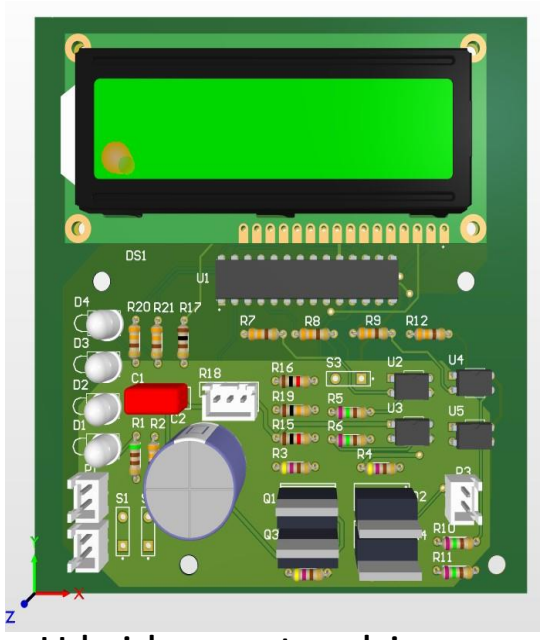
Tharindu Samarakoon
Dept. of Electronic & Telecommunication Eng., University of Moratuwa

09th October 2021

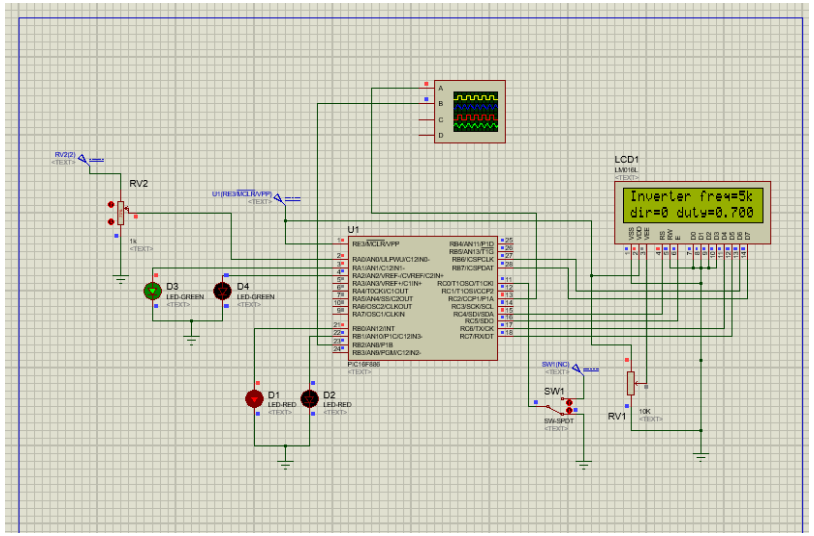
Today we do…

- Program the H bridge motor driver circuit
- Simulate the design
- Test on a demonstration board





H bridge motor driver



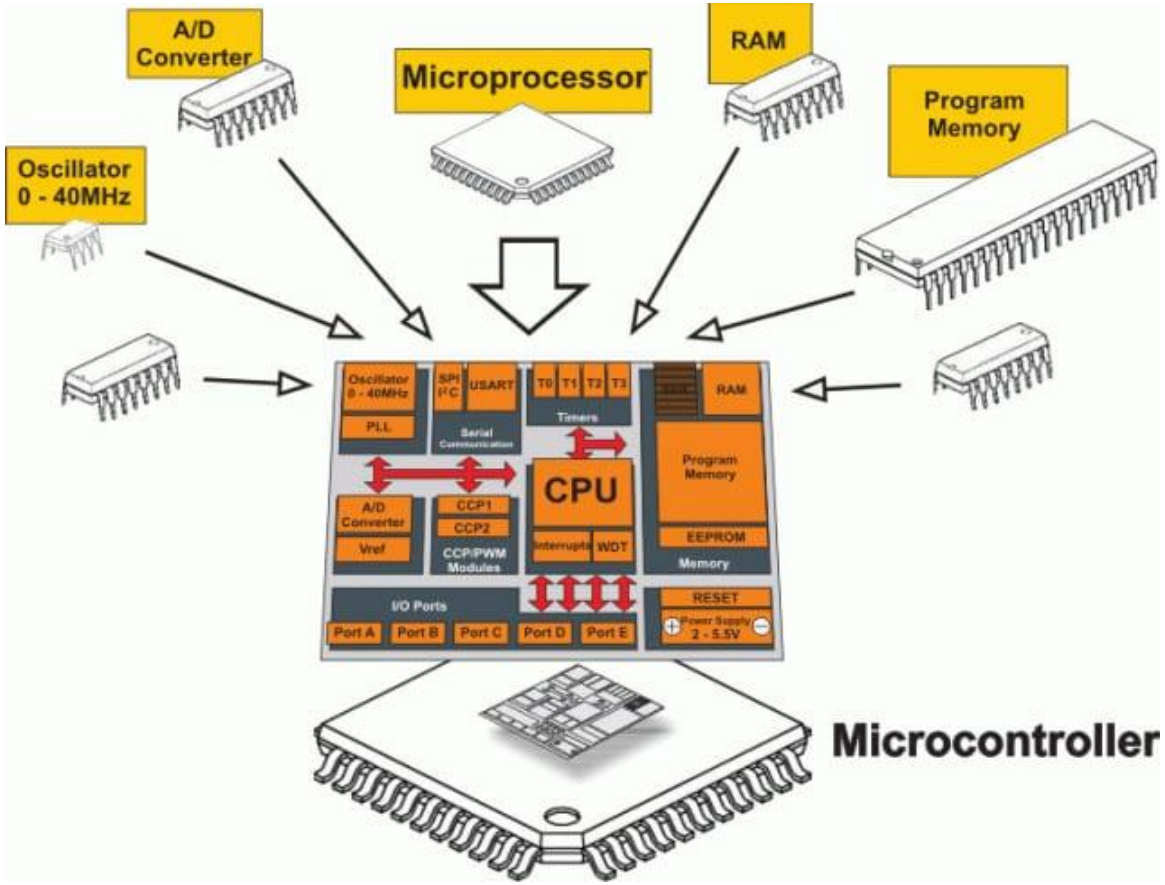
Proteus simulation

Compliment PWM signal output

What is an MCU

A small computer on a single integrated circuit.

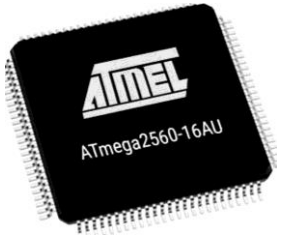
- Processor core
- Storage
  - RAM – data storage
  - ROM, EPROM, Flash – program storage
- Programmable GPIOs
- Communication interfaces – USART, I2C, SPI
- Peripherals – timers, PWM generators, internal oscillators, ADC units



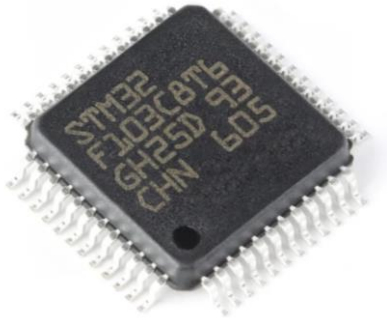
Popular microcontrollers

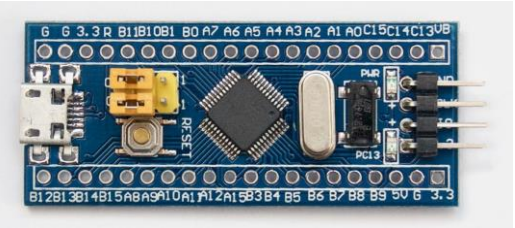
- PIC microcontrollers

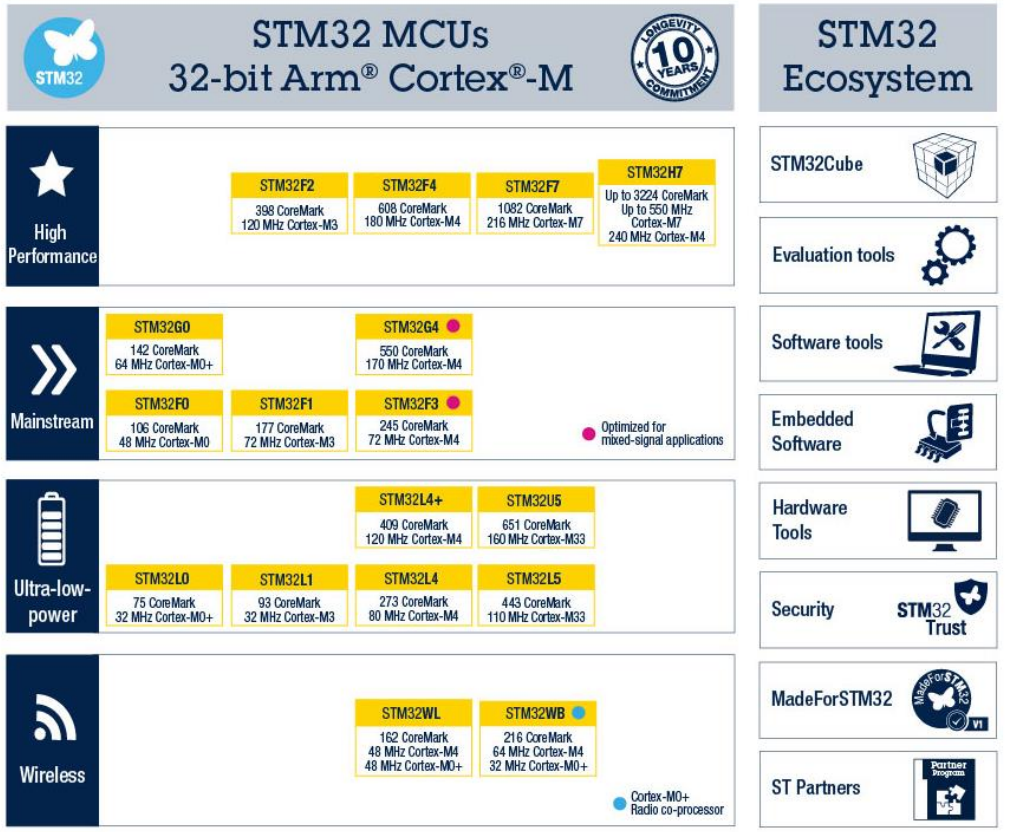
- AVR microcontrollers
  - Arduino boards



- STM32 microcontrollers
  - BluePill STM32
  - High performances
  - Large community
  - Many resources

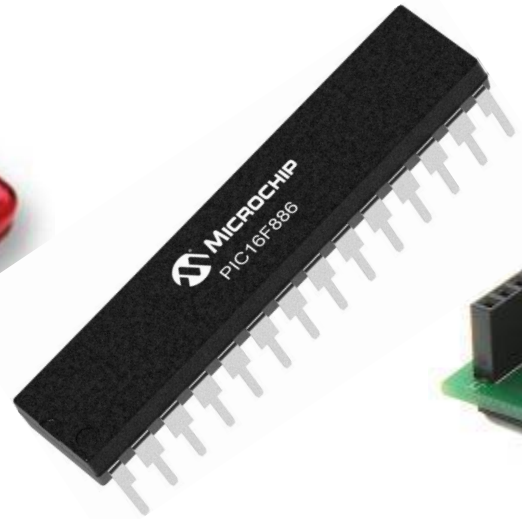


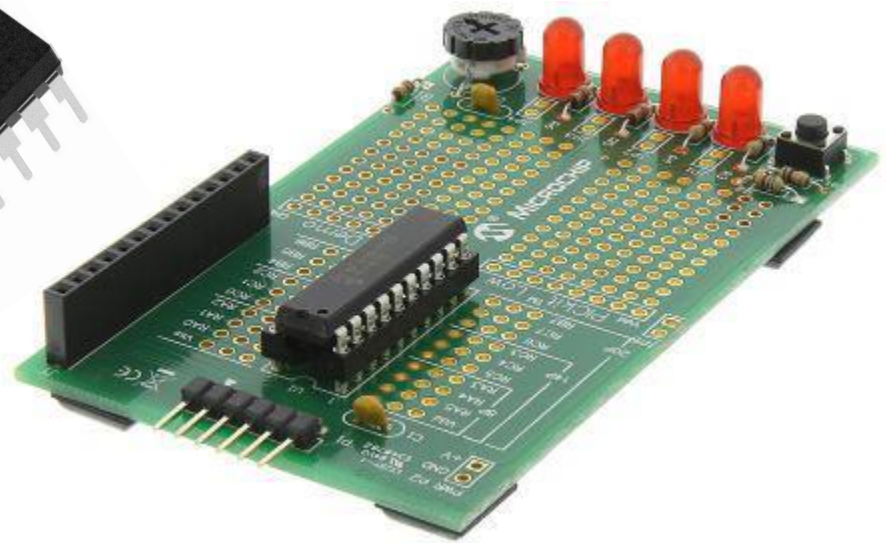




Required software & devices

- MPLAB X IDE
- xc8 – c compiler
- Proteus Design Suit
- PICkit3 programmer
- PIC16f886 MCU





PIC 28 Pin demo board

Design files:-
https://github.com/tharinduSamare/PIC_workshop.git

PIC16F886 datasheet

High-Performance RISC CPU:

- Only 35 instructions to learn:
  - All single-cycle instructions except branches
- Operating speed:
  - DC – 20 MHz oscillator/clock input
  - DC – 200 ns instruction cycle
- Interrupt capability
- 8-level deep hardware stack
- Direct, Indirect and Relative Addressing modes

Special Microcontroller Features:

- Precision Internal Oscillator:
  - Factory calibrated to ±1%
  - Software selectable frequency range of 8 MHz to 31 kHz
  - Software tunable
  - Two-Speed Start-up mode
  - Crystal fail detect for critical applications
  - Clock mode switching during operation for power savings
- Power-Saving Sleep mode
- Wide operating voltage range (2.0V-5.5V)
- Industrial and Extended Temperature range
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Brown-out Reset (BOR) with software control

Peripheral Features:

- 24/35 I/O pins with individual direction control:
  - High current source/sink for direct LED drive
  - Interrupt-on-Change pin
  - Individually programmable weak pull-ups
  - Ultra Low-Power Wake-up (ULPWU)
- Analog Comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (CV_{REF}) module (% of V_{DD})
  - Fixed voltage reference (0.6V)
  - Comparator inputs and outputs externally accessible
  - SR Latch mode
  - External Timer1 Gate (count enable)
- A/D Converter:
  - 10-bit resolution and 11/14 channels
- Timer0: 8-bit timer/counter with 8-bit programmable prescaler
- Enhanced Timer1:
  - 16-bit timer/counter with prescaler
  - External Gate Input mode
  - Dedicated low-power 32 kHz oscillator
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Enhanced Capture, Compare, PWM+ module:
  - 16-bit Capture, max. resolution 12.5 ns
  - Compare, max. resolution 200 ns
  - 10-bit PWM with 1, 2 or 4 output channels, programmable “dead time”, max. frequency 20 kHz
  - PWM output steering control

PIC16F886 datasheet

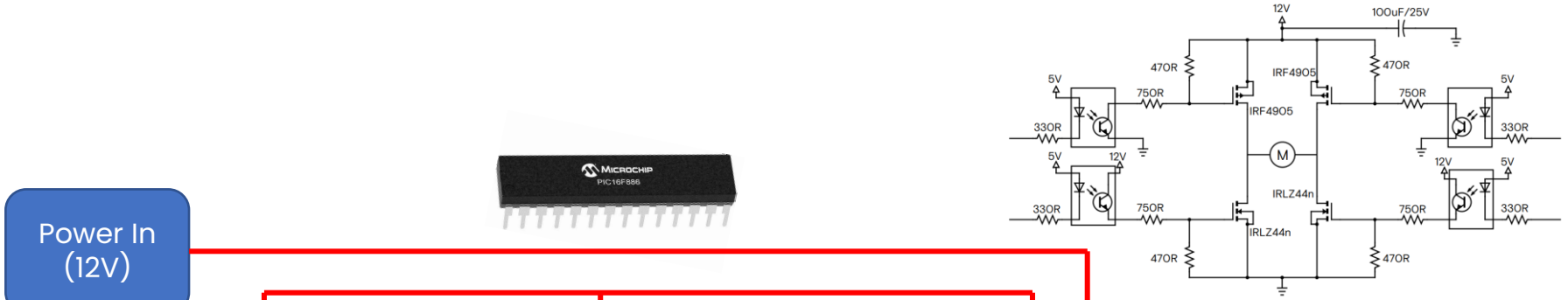
Special Function Registers

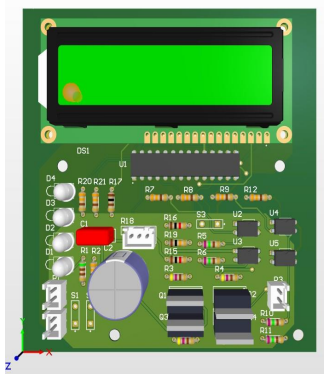
- GPIOs
  - TRISA, TRISB … :- Decides input / output
  - PORTA, PORTB … :- Value of the pin (1/0)
  - ANSEL, ANSELH :- Pin type (Analog / digital)
- PWM signals
  - CCP1CON – PWM type
  - CCPR1L – PWM duty cycle
  - PR2 – PWM period
  - T2CON – Timer_2 configurations
- ADC unit
  - ADCON0 – ADC configurations
  - ANSEL, ANSELH – Pin type(analog / digital)

FIGURE 2-6: PIC16F886/PIC16F887 SPECIAL FUNCTION REGISTERS

	File Address		File Address		File Address		File Address
Indirect addr. (1)	00h	Indirect addr. (1)	80h	Indirect addr. (1)	100h	Indirect addr. (1)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h	WDTCON	105h	SRCON	185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h	CM1CON0	107h	BAUDCTL	187h
PORTD(2)	08h	TRISD(2)	88h	CM2CON0	108h	ANSEL	188h
PORTE	09h	TRISE	89h	CM2CON1	109h	ANSELH	189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDAT	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2(1)	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved	18Eh
TMR1H	0Fh	OSCCON	8Fh	EEADRH	10Fh	Reserved	18Fh
T1CON	10h	OSCTUNE	90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h	WPUB	95h		115h		195h
CCPR1H	16h	IOCB	96h	General Purpose Registers 16 Bytes	116h	General Purpose Registers 16 Bytes	196h
CCP1CON	17h	VRCON	97h		117h		197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah	SPBRGH	9Ah		11Ah		19Ah
CCPR2L	1Bh	PWM1CON	9Bh		11Bh		19Bh
CCPR2H	1Ch	ECCPAS	9Ch		11Ch		19Ch
CCP2CON	1Dh	PSTRCON	9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
General Purpose Registers 96 Bytes	20h – 6Fh	General Purpose Registers 80 Bytes	A0h – EFh	General Purpose Registers 80 Bytes	120h – 16Fh	General Purpose Registers 80 Bytes	1A0h – 1EFh
	70h – 7Fh	accesses 70h-7Fh	F0h – FFh	accesses 70h-7Fh	170h – 17Fh	accesses 70h-7Fh	1F0h – 1FFh
Bank 0		Bank 1		Bank 2		Bank 3	

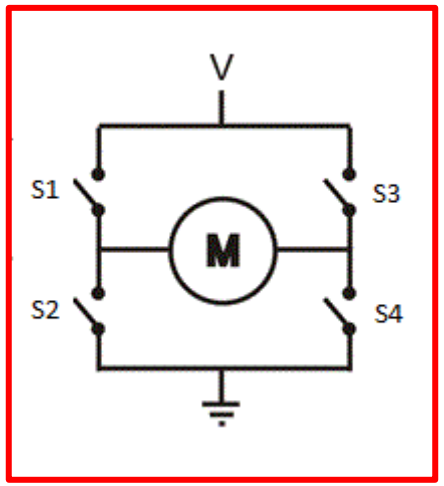
Design Block diagram

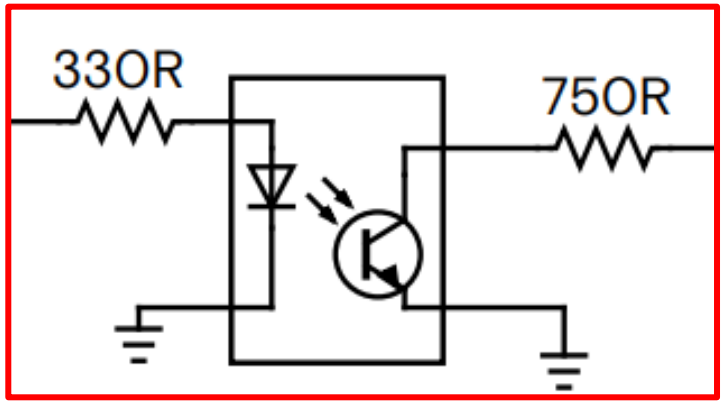




H bridge motor driver

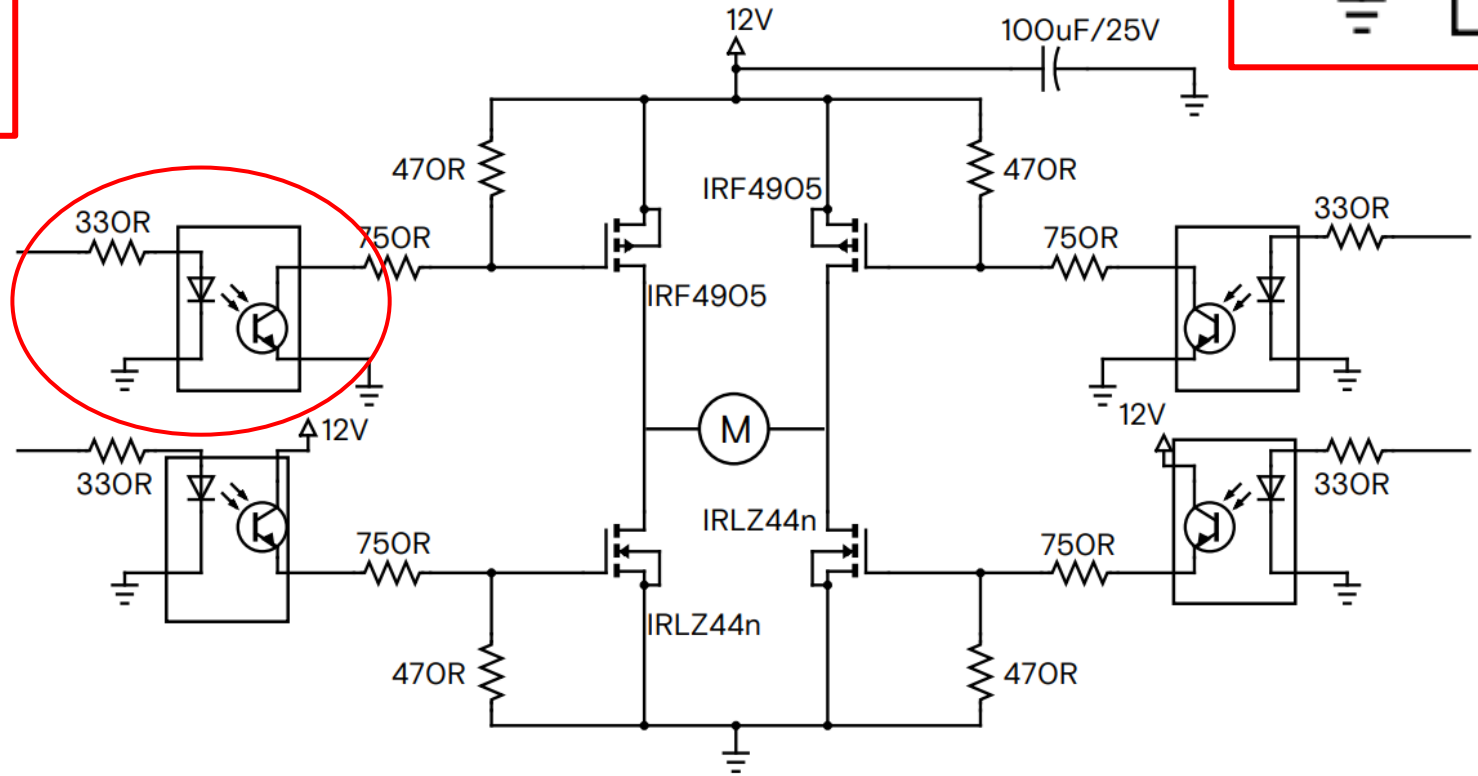
H bridge control



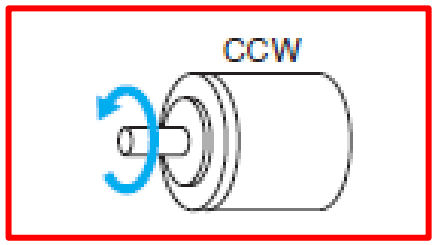


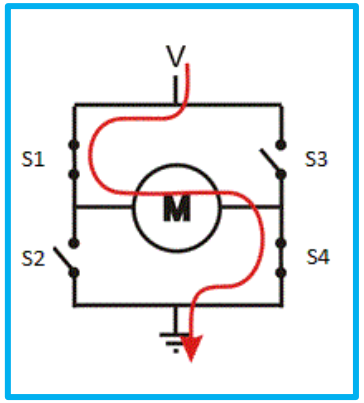
DC signal

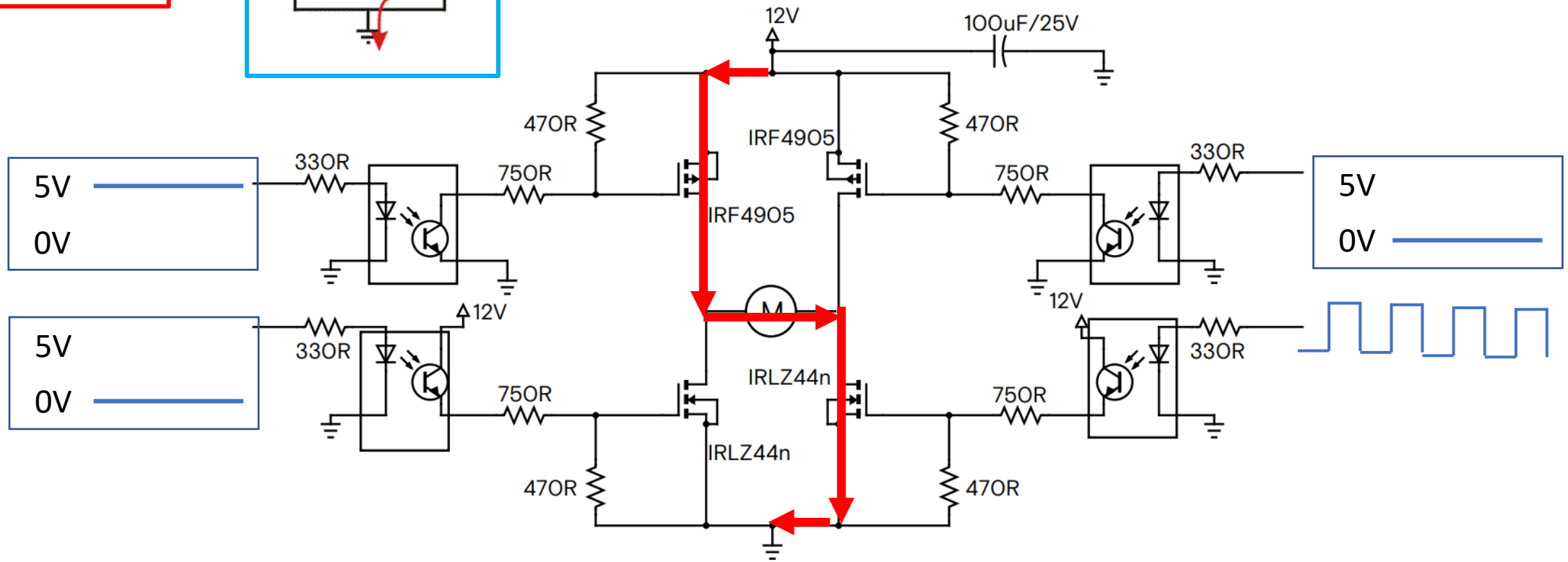
PWM signal



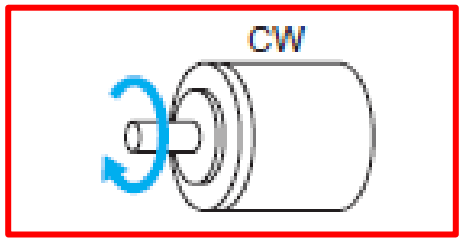
H bridge control

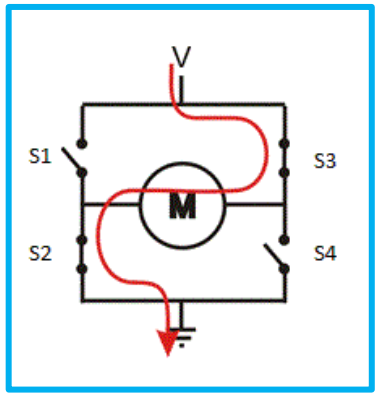


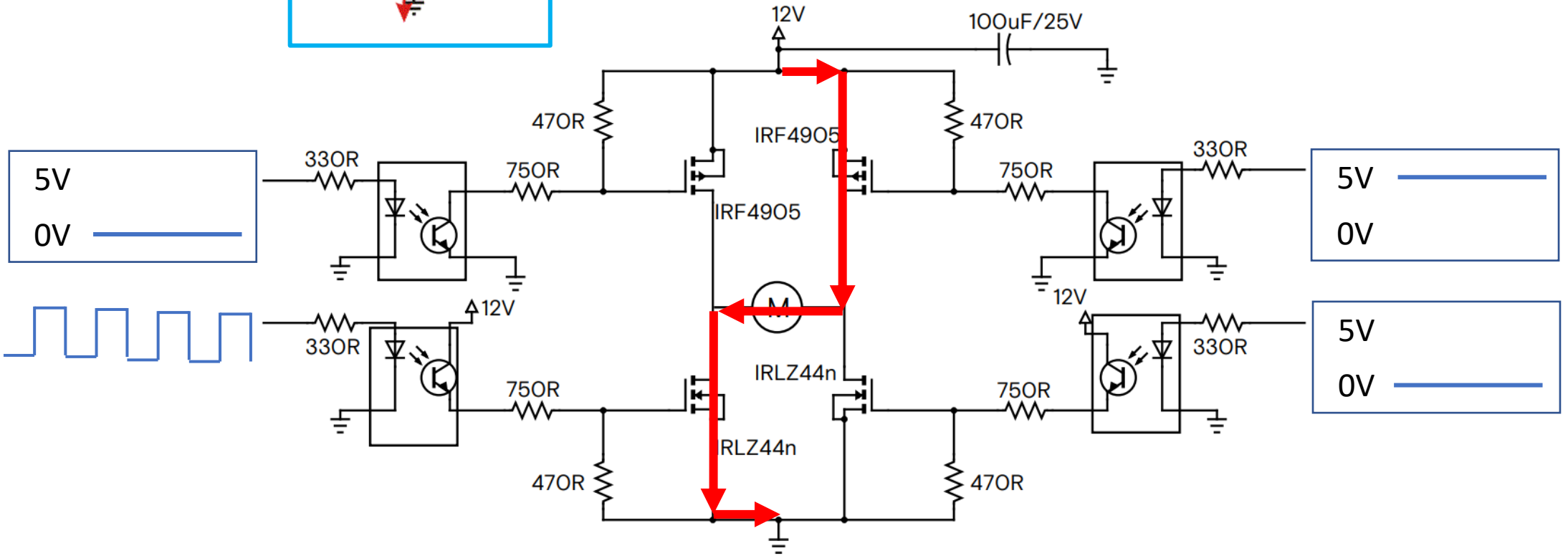




H bridge control







C coding

- Insert Libraries
  - Include <stdint.h>
  - include “lcd.h”
- #define directive
  - `#define LED0 PORTBbits.RB0`
- Set or clear a register bit
  - PORTAbits.RA3 = 1;
  - PORTAbits.RA3 = 0;
  - RA3 = 1;
  - RA3 = 0;
  - PORTA = PORTA | (1 << 3); // 0b01000010 | 0b00001000
  - PORTA = PORTA & ~(1<<3); // 0b01001010 & ~(0b00001000) // 0b01001010 & 0b11110111
- Set or clear a full register
  - PORTA = 0b10010011;
  - PORTA = 0x93;

C coding

- Data types
  - Int, float …
  - uint8_t, uint16_t …
  - bool
- Functions
  - Function declaration
  - Function body
  - Function call
- Built-in functions

```
uint16_t adc_value;
```

```
bool current_direction = 0;
```

```
uint16_t ADC_Read(uint8_t channel);
```

```
uint16_t ADC_Read(uint8_t channel){
  ADCON0 &= 0x11000101; //Clearing the Channel Selection Bits
  ADCON0 |= (unsigned)channel<<3; //Setting the required Bits
  __delay_ms(2); //Acquisition time to charge hold capacitor
  GO_nDONE = 1; //Initializes A/D Conversion
  while(GO_nDONE); //Wait for A/D Conversion to complete
  return (uint16_t)((ADRESH<<8)+ADRESL); //Returns Result
}
```

```
adc_value = ADC_Read(0);
```

```
  __delay_ms(50);
```

C coding

- If conditions

```
if (PORTAbits.RA1 == 1){
    PORTBbits.RB1 = 1;
}
else{
    PORTBbits.RB1 = 0;
}
```

- For loops

```
for (uint8_t i = 0; i< 8;i++){
    if (adc_value > ADC_limits[i]){
        PORTB = PORTB | (1U<<i); // LED on
    }
    else{
        PORTB = PORTB & ~(1U<<i) ; // LED off
    }
}
```

- While loops

```
while (1){
  adc_value = ADC_Read(0);
  if (adc_value != pulse_width){
      pulse_width = adc_value;
      set_PWM_duty_cycle(pulse_width);
  }
  __delay_ms(50);
}
```

Coding

1. Set configuration bits
2. Include libraries
3. Set oscillator frequency
   - useful for delay functions
4. Define aliases
5. Define global variables
6. Function prototypes
7. Main function
   - Initialize the system
   - Infinite loop
8. Other functions

```
// CONFIG1
#pragma config FOSC = INTRC_NOCLKOUT//
#pragma config WDTE = OFF       // Watc
#pragma config PWRTE = OFF      // Powe
#pragma config MCLRE = ON       // RE3/
#pragma config CP = OFF         // Code
#pragma config CPD = OFF        // Data
#pragma config BOREN = OFF      // Brow
#pragma config IESO = OFF       // Inte
#pragma config FCMEN = OFF      // Fail
#pragma config LVP = ON         // Low

// CONFIG2
#pragma config BOR4V = BOR40V   // Brow
#pragma config WRT = OFF        // Flas
```

```
#include <xc.h>
#include "config.h"
#include <stdint.h>
#include "pic16f886.h"
#include <stdio.h>
#include <stdbool.h>
```

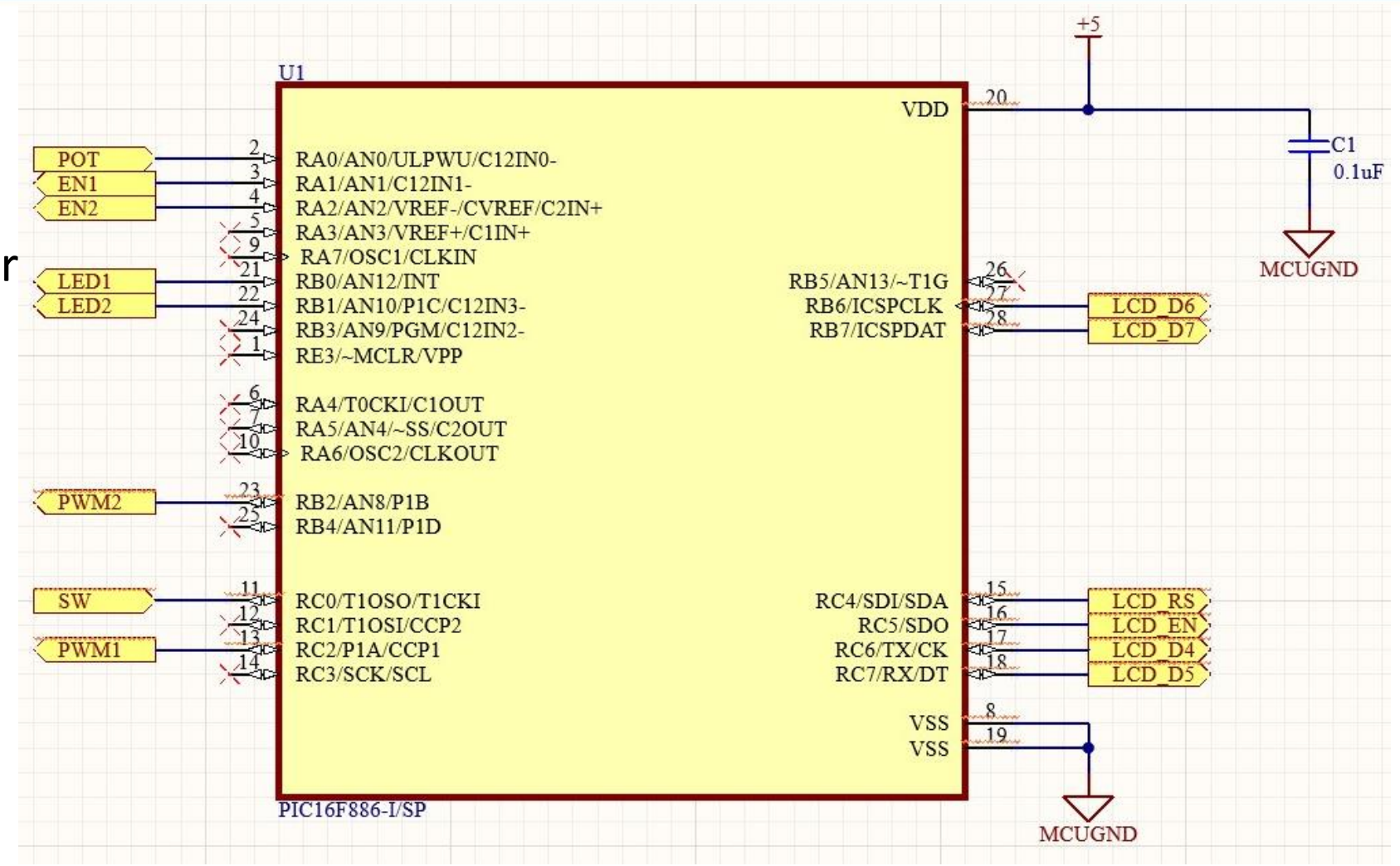
```
#define _XTAL_FREQ 4000000
```

```
#define LED0 PORTBbits.RB0
#define LED1 PORTBbits.RB1
#define direction_0 RA1
#define direction_1 RA2
```

```
void PWM_Initialize(long PWM_freq, uint16_t init_pulse_width );
void set_PWM_duty_cycle(uint16_t duty);
void ADC_Initialize();
uint16_t ADC_Read(uint8_t channel);
```

Design + simulation

1. Analog input signal capture
2. PWM signal output
3. LCD display
4. Final design - H bridge driver

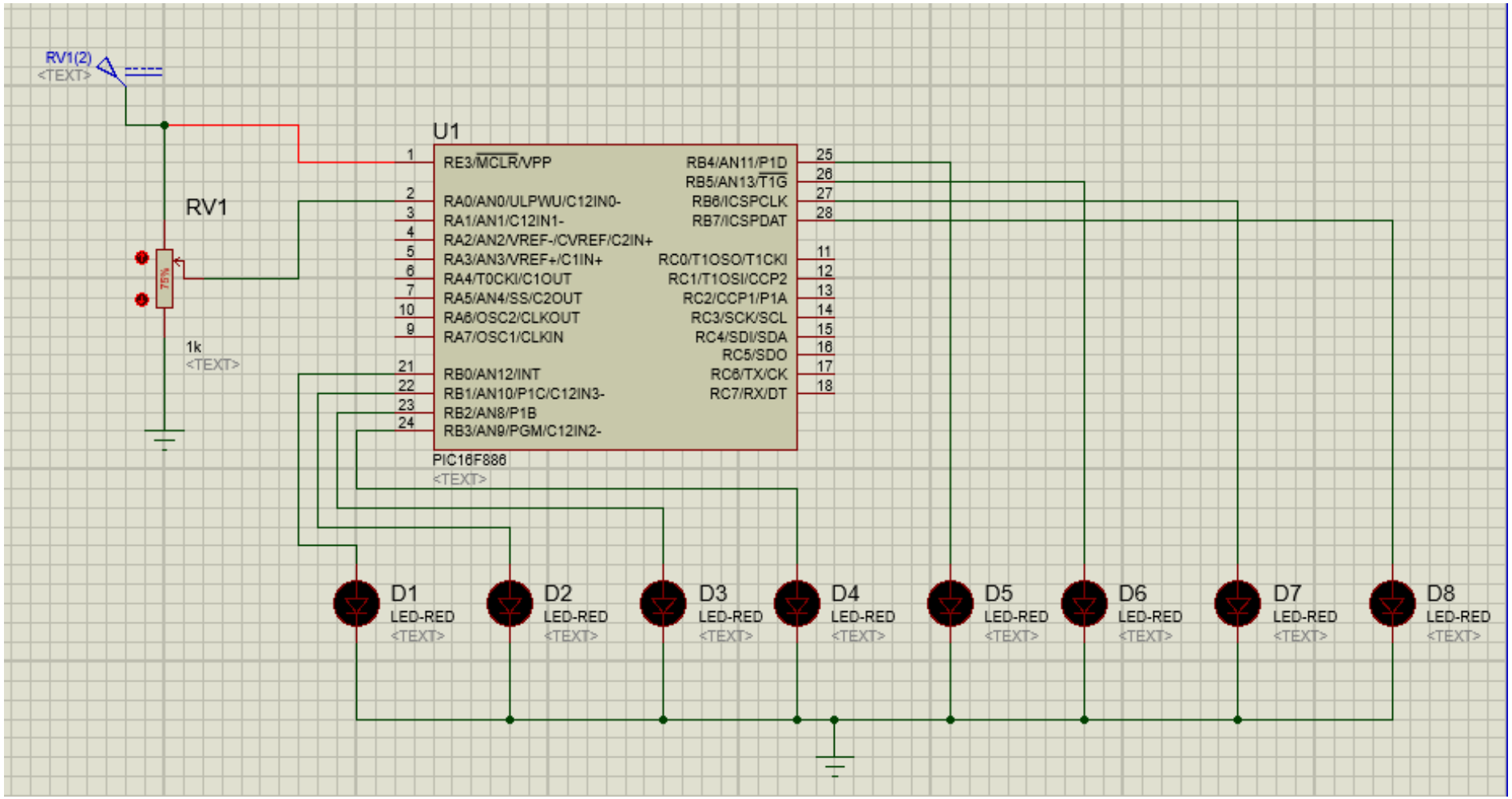


Pin connection diagram

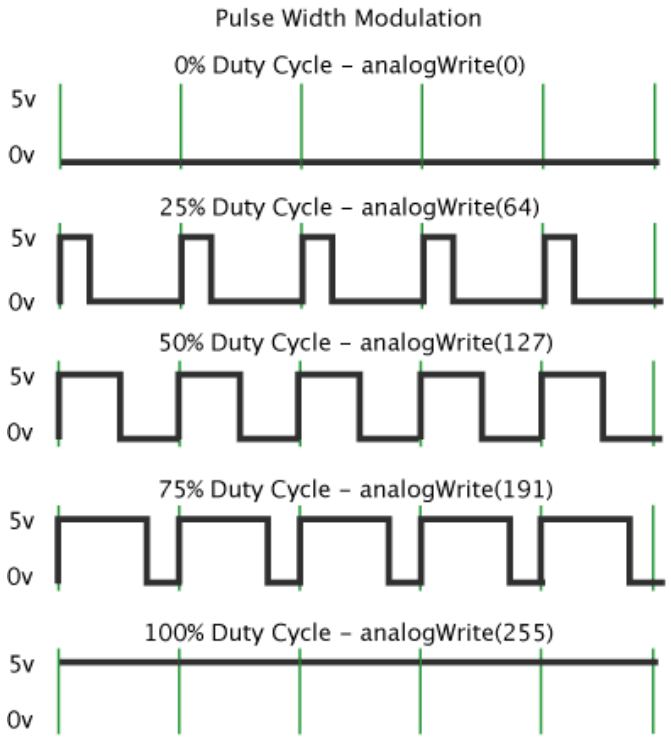
Design files:-
https://github.com/tharinduSamare/PIC_workshop.git

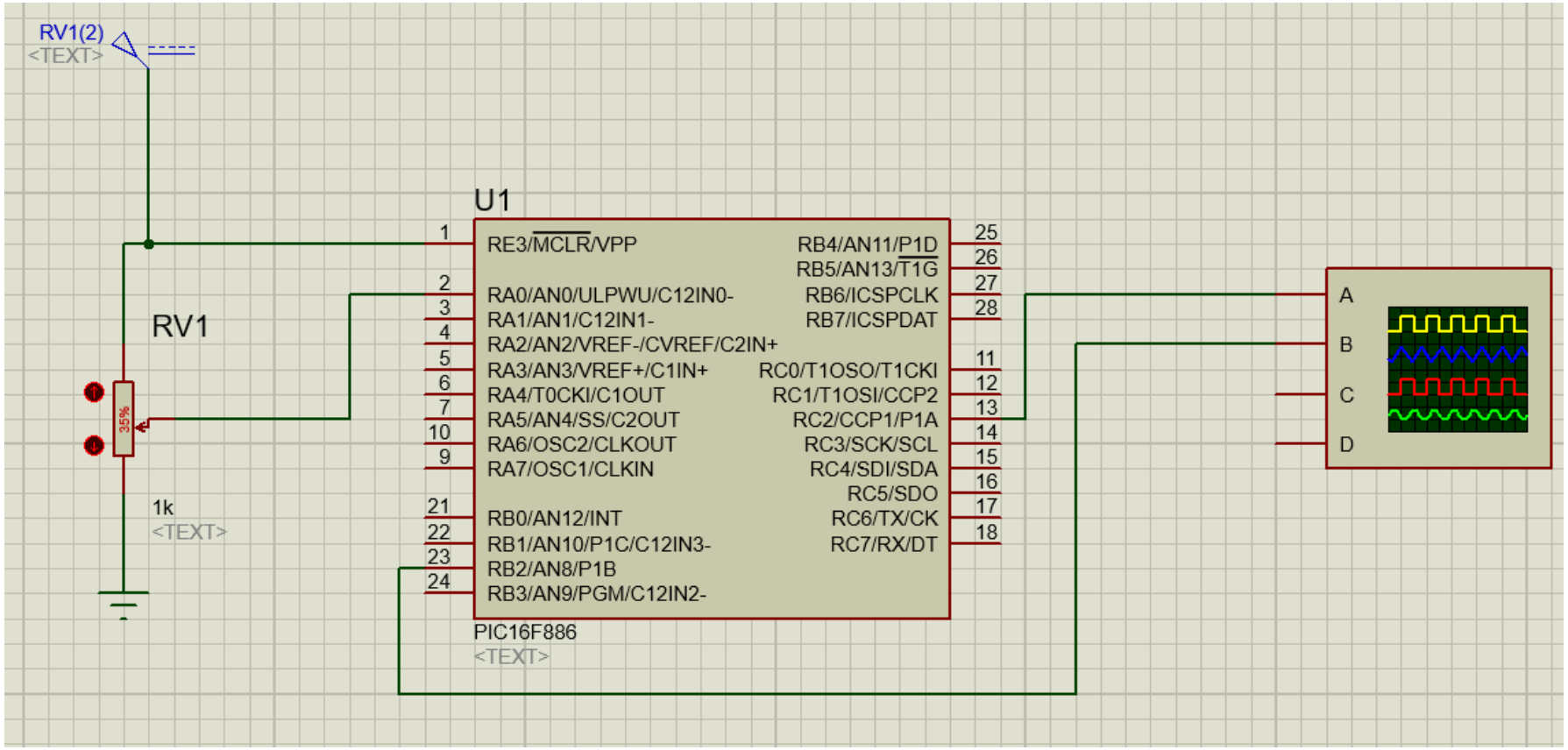
Analog input capture

- GPIOs
  - TRISA,
  - PORTA, PORTB
  - ANSEL, ANSELH
- ADC unit
  - ADCON0
  - ANSEL, ANSELH



PWM signal generate

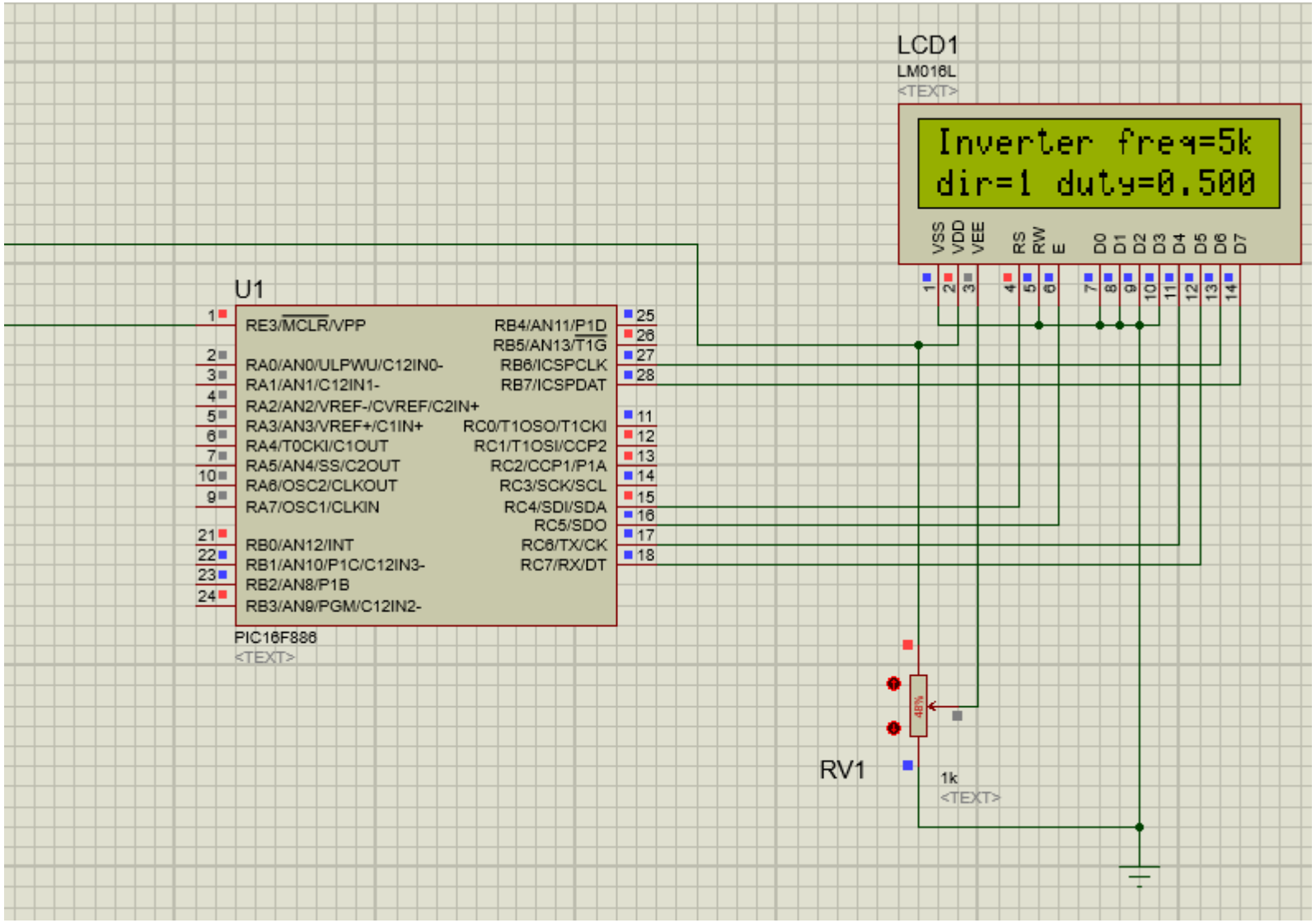




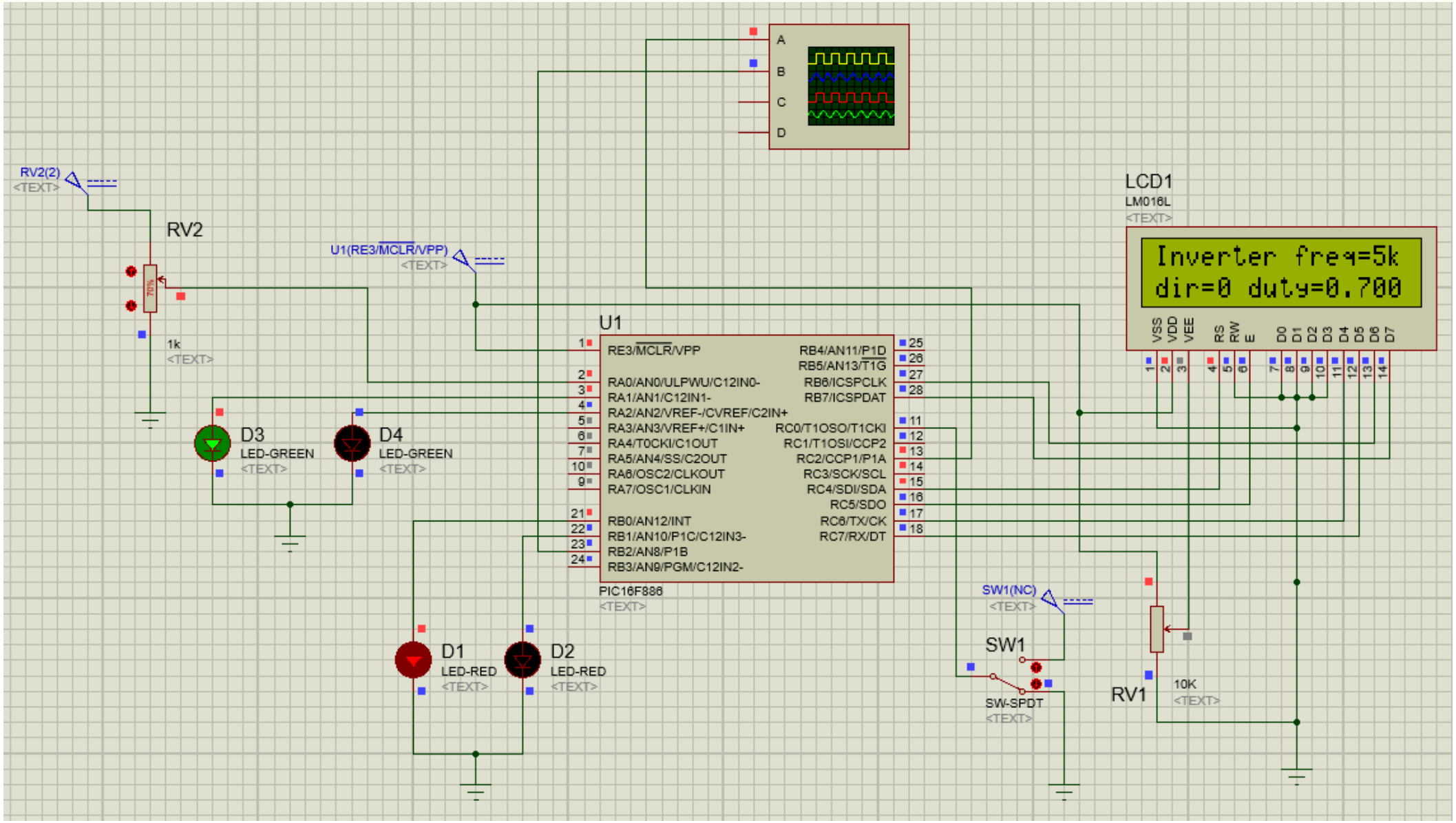
- ADC unit
  - PR2
  - T2CON
  - CCP1CON
  - CCPR1L
- GPIOs
  - TRISA, TRISB, TRISC
  - PORTA, PORTB, PORTC
  - ANSEL, ANSELH

LCD display

- GPIOs
  - TRISA, TRISB, TRISC
  - PORTA, PORTB, PORTC
  - ANSEL, ANSELH

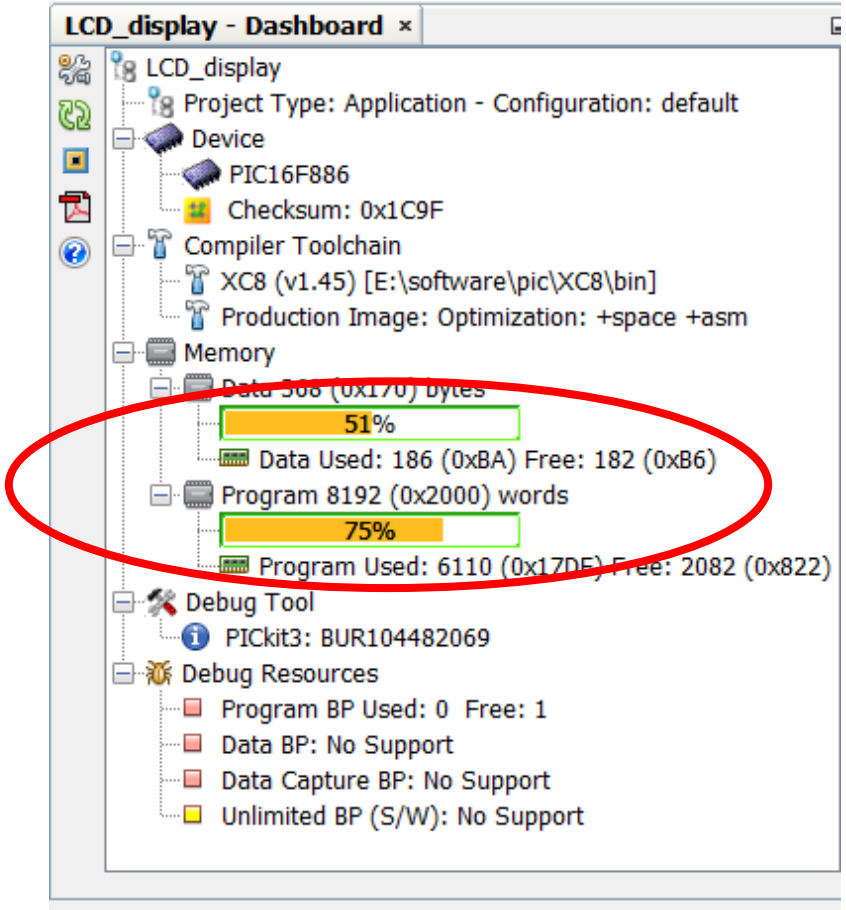


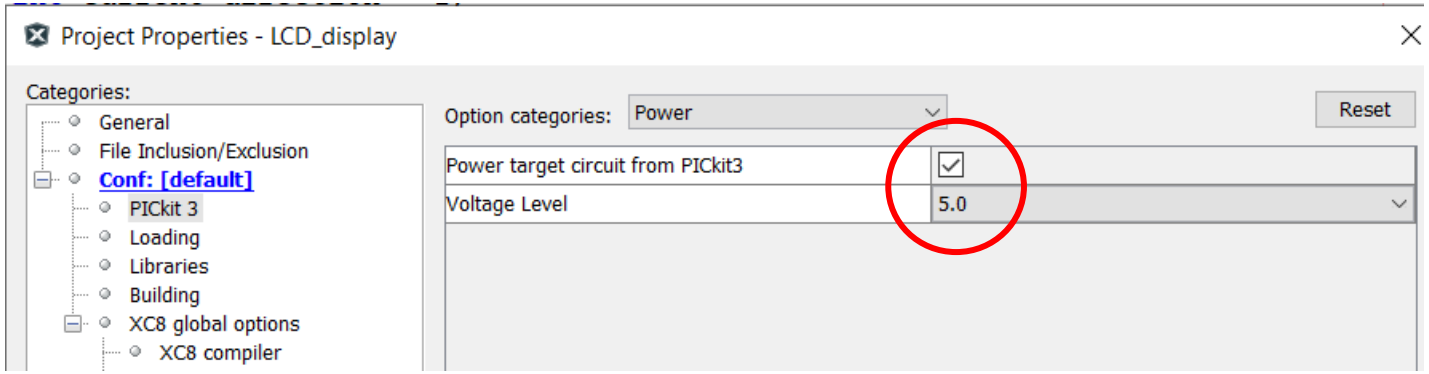
H bridge driver



Program with PICkit3

- See the memory usage
- Uploading methods
  - Using MPLAB X IDE
  - Using PICkit3 software
- Supply power to the MCU from PICkit

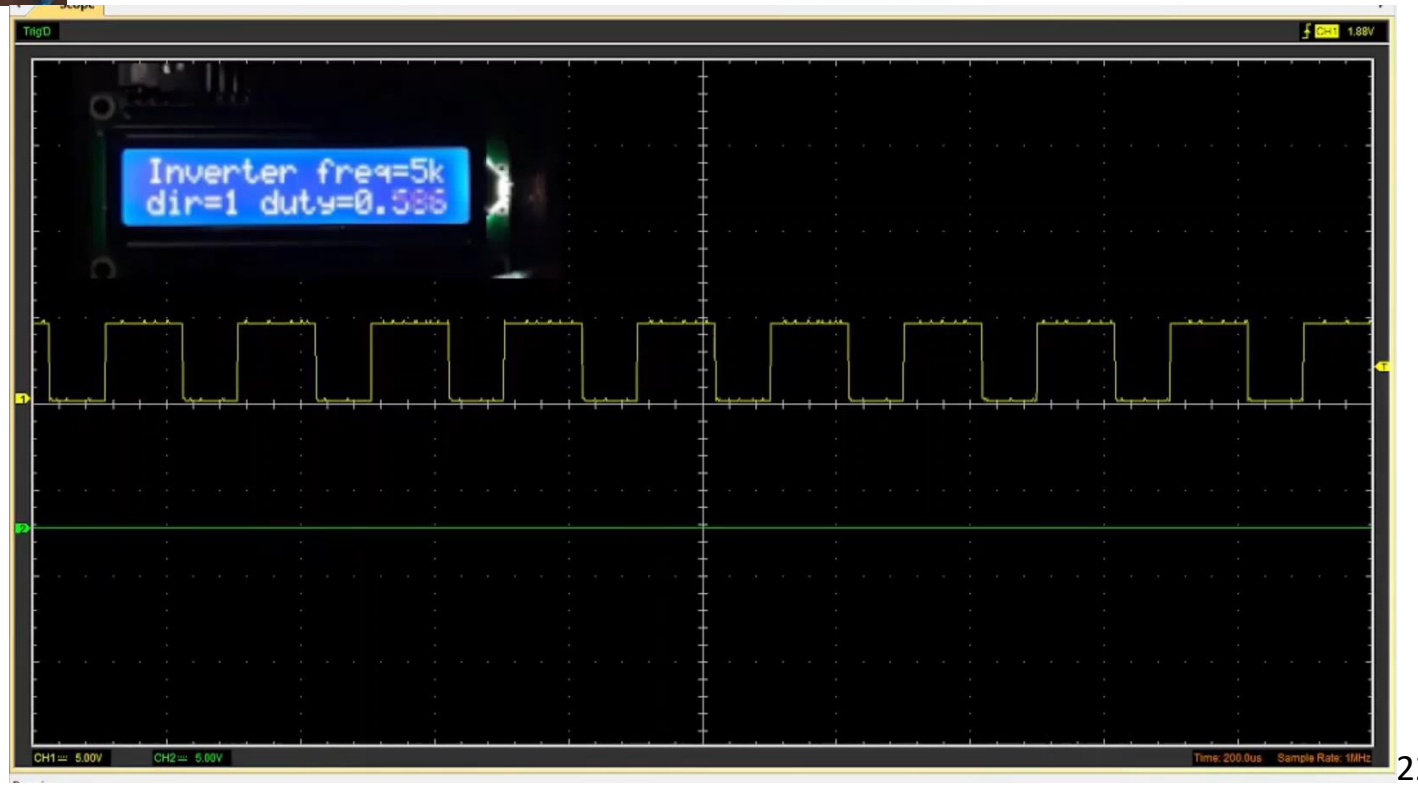




Final output

Implementation on a demo board

Oscilloscope output



Thank you!