Introduction to IoT
LAB Exercises

Karel Slavicek
Vaclav Oujezsky

2024

IoT LAB - Outline

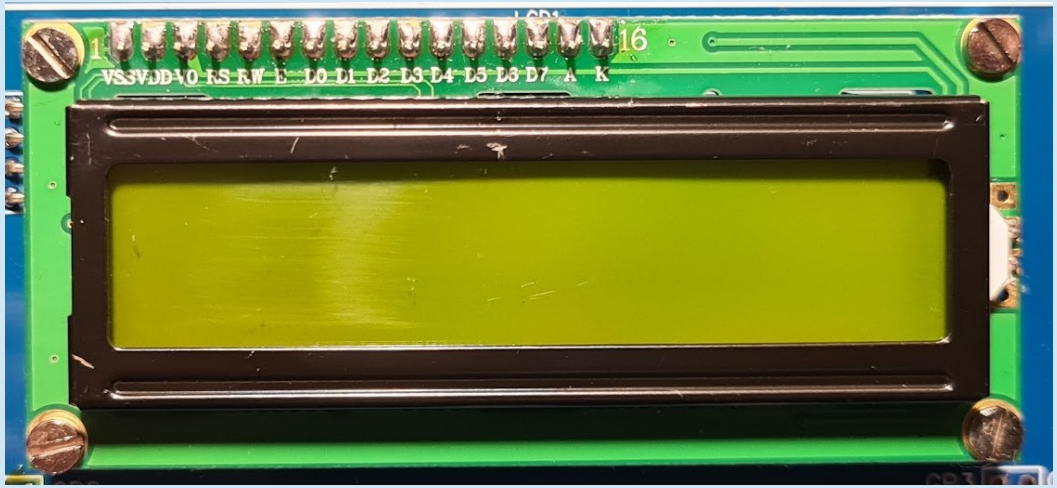
- Rotary encoder
- PWM modulation (if can fit in)

Hardware Overview

- MCU board - STM32
- Communication board – CANbus
- Application board - LCD

LCD

- Liquid Crystal Display
- Invented in 1964
- Used to display information
- Various modes of operation
- Used in
  - Watches
  - Laptops
  - Phones



Using LCD

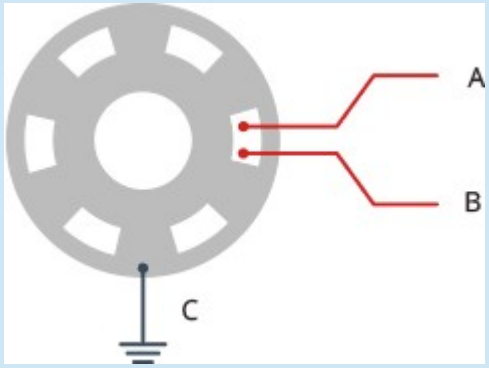
- Libraries
  - https://github.com/johnrickman/LiquidCrystal_I2C/tree/c976ad2dc682d40d994797761f39d4d9f0cc8196
- Include the libraries
  - Wire.h to use I2C
  - `#include <Wire.h>`
  - `#include <LiquidCrystal_I2C.h>`
- Initialize LCD
  - 16 x 2 LCD
  - Using 0x3F I2C address
  - `LiquidCrystal_I2C oLcd(0x3F, 16, 2);`
  - `oLcd.init();`
  - `oLcd.backlight();`
  - `oLcd.setCursor(0, 0);`
- Functions available
  - `oLcd.clear();`
  - `oLcd.print("Hello world");`
  - `oLcd.setCursor(0, 1);`

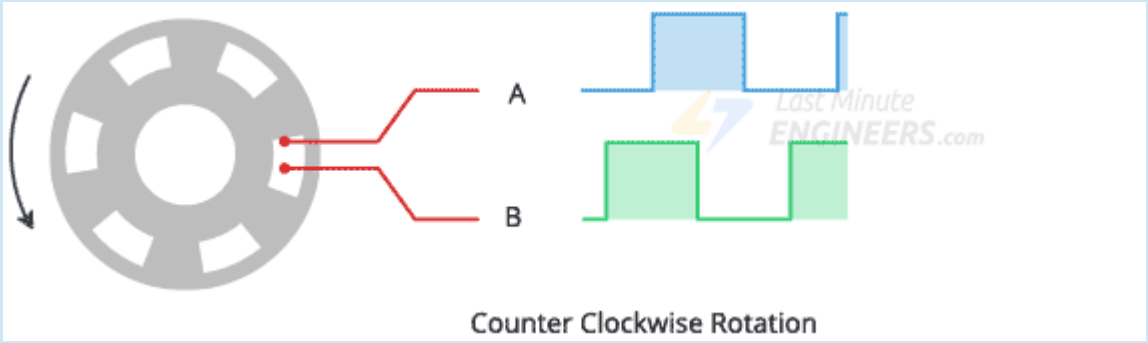
LCD

- Serves as supportive tool today, we focus on the rotary encoder
- Will be discuse in more detailes later

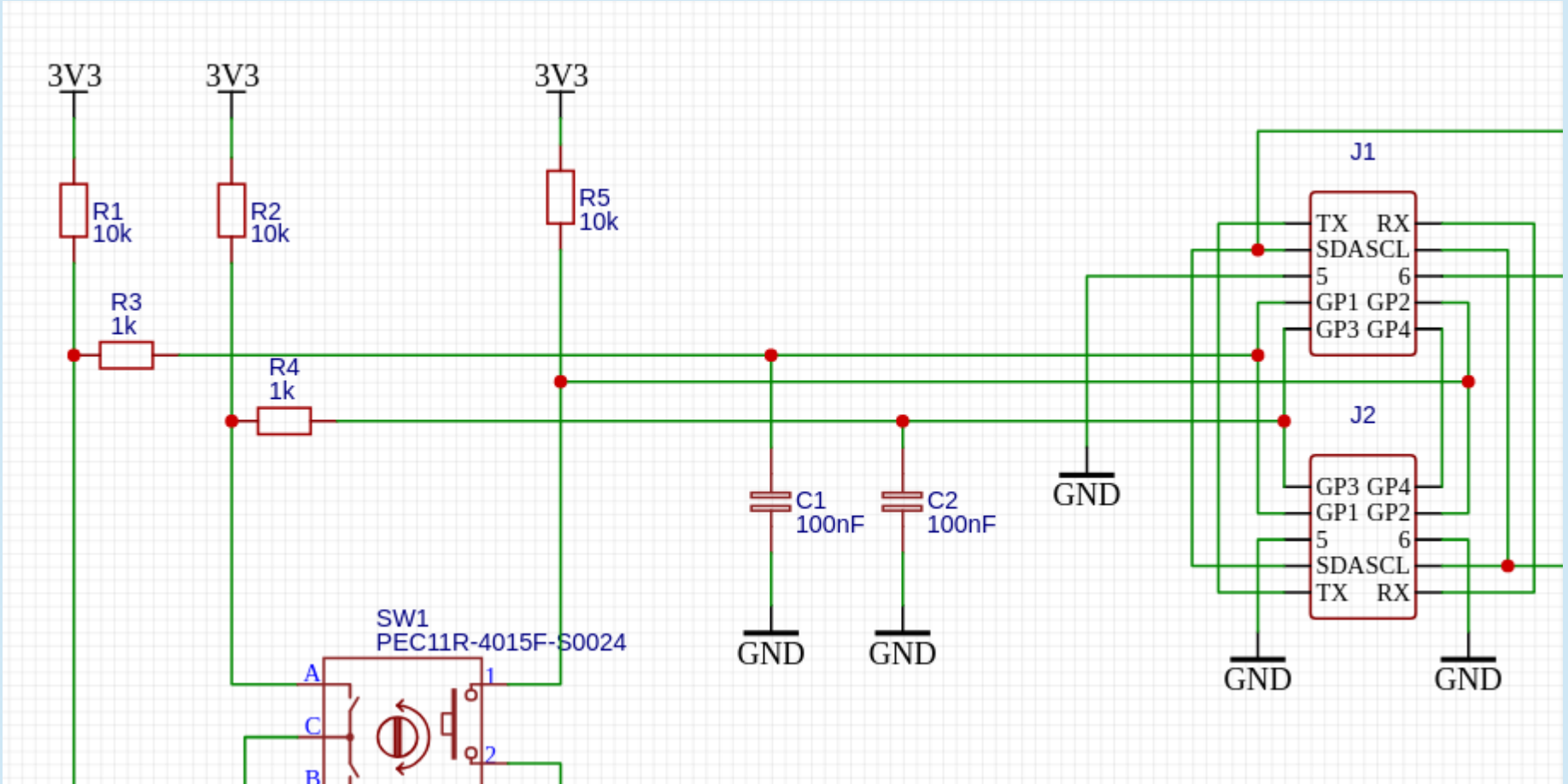
Rotary encoder – how it works

- When you turn the knob, A and B make contact with the common ground pin C in a specific order





HW structure



STM32 pinout

STM32 BluePill

IoT LAB	BluePill	IoT LAB	BluePill
UART TX	PA2 (UART2)	**SPI CS**	PA4 (SPI 1)
UART RX	PA3 (UART2)	**SPI SCK**	PA5 (SPI 1)
I2C SCL	PB6 (I2C 1)	**SPI MISO**	PA6 (SPI 1)
I2C SDA	PB7 (I2C 1)	**SPI MOSI**	PA7 (SPI 1)
GP-1	PB5	**GP-5**	PB15
GP-2	PA15	**GP-6**	PB13
GP-3	PB4	**GP-7**	PB14
GP-4	PB3	**GP-8**	PB12

Rotary encoder pinout

- #define CLK PB4
- #define DT PB5
- #define BTN PA15

Using rotary encoder

```
void setup() {

// Set encoder pins as inputs
pinMode(CLK,INPUT);
pinMode(DT,INPUT);
pinMode(SW, INPUT_PULLUP);

// Setup Serial Monitor
Serial.begin(9600);

// Read the initial state of CLK
lastStateCLK = digitalRead(CLK);
}
```

Using rotary encoder

```
void loop() {

// Read the current state of CLK
currentStateCLK = digitalRead(CLK);
// If last and current state of CLK are different, then pulse occurred
if (currentStateCLK != lastStateCLK  && currentStateCLK == 1){

// If the DT state is different than the CLK state then the encoder is rotating CCW so decrement
if (digitalRead(DT) != currentStateCLK) {
counter --;
currentDir ="CCW";
} else {
counter ++;
currentDir ="CW";
}
Serial.print("Counter: ");
Serial.println(counter);
}
```

Using rotary encoder

```
// Remember last CLK state
lastStateCLK = currentStateCLK;

// Read the button state
int btnState = digitalRead(SW);

if (btnState == LOW) {
if (millis() - lastButtonPress > 50) {
Serial.println("Button pressed!");
}

// Remember last button press event
lastButtonPress = millis();
}

delay(1);
}
```

Interrupts

- Why do we need interrupts?
- How it works?

Rotary encoder with interrupts

```
int counter = 0;
int currentStateCLK;
int lastStateCLK;
String currentDir = "";
unsigned long lastButtonPress = 0;

void setup() {
  pinMode(CLK, INPUT);
  pinMode(DT, INPUT);
  pinMode(BTN, INPUT_PULLUP);

  lastStateCLK = digitalRead(CLK);

  oLcd.init();
  oLcd.backlight();
  oLcd.setCursor(0, 0);

  attachInterrupt(CLK, updateEncoder, CHANGE);
  attachInterrupt(DT, updateEncoder, CHANGE);
  attachInterrupt(BTN, updateBtnPressed, CHANGE);
}
```

Rotary encoder with interrupts

```
void updateEncoder(){
  currentStateCLK = digitalRead(CLK);

  if (currentStateCLK != lastStateCLK  && currentStateCLK == 1){

    if (digitalRead(DT) != currentStateCLK) {
      counter --;
      currentDir ="CCW";
    } else {
      counter ++;
      currentDir ="CW";
    }
    oLcd.clear();
    oLcd.print("Direction: ");
    oLcd.print(currentDir);
    oLcd.setCursor(0, 1);
    oLcd.print("Counter: ");
    oLcd.print(counter);
  }

  lastStateCLK = currentStateCLK;
}
```

Now it is the time for your own experiments!