A small diversion from Beginners Tutorial (for those who wants some difficult stuff). Lets try something more creative with our OUTPUT I/O Pins and our simple digitalWrite() function which previously was just for turning LED ON and OFF

By supplying a series of 0V or 5V to OUTPUT I/O Pins at a specific sequence, we will be able to display things on the i2c SSD1306 OLED Screen attached to those OUTPUT I/O Pins
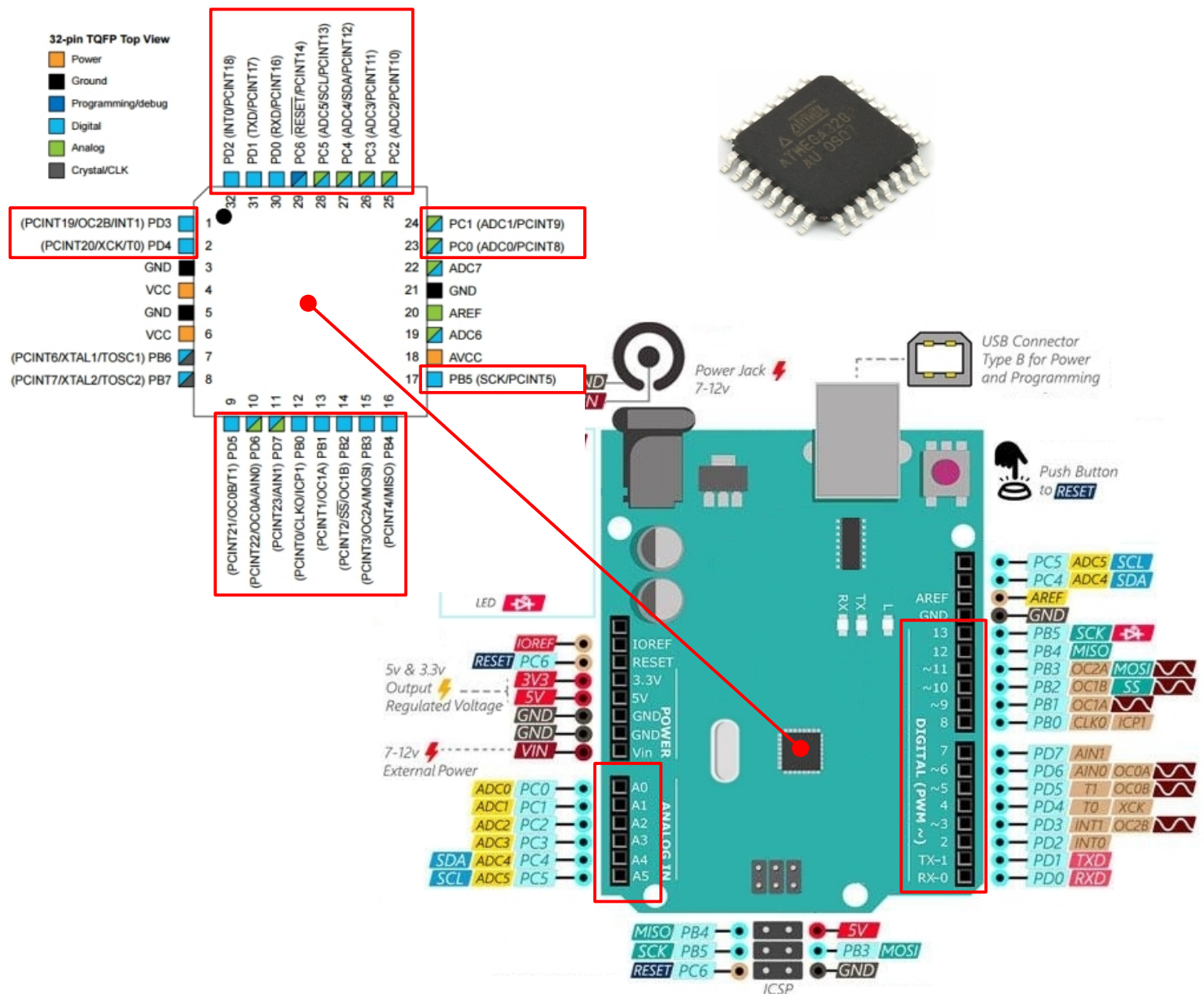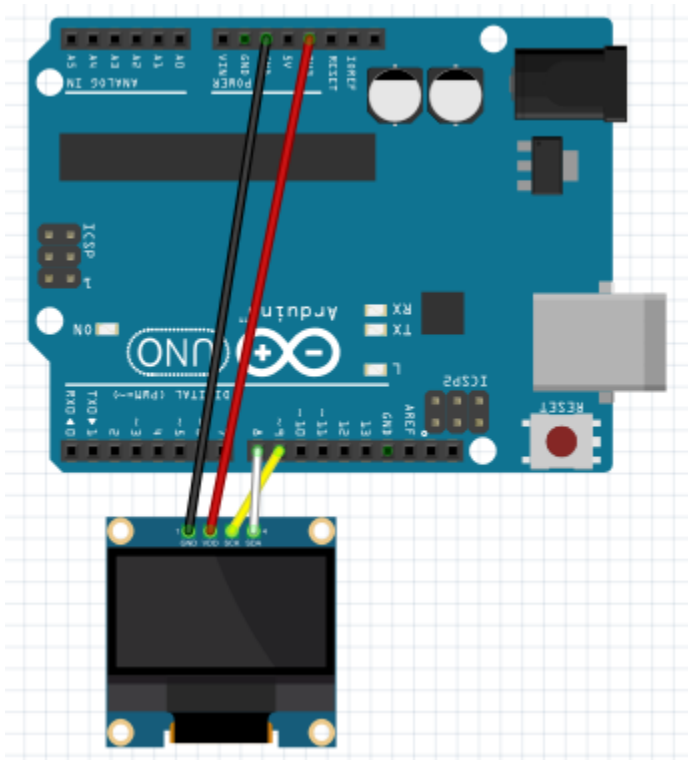

Below is the i2c SSD1306 OLED Screen 64x128 Pixel. IF you do not have this device, just look at the source codes.
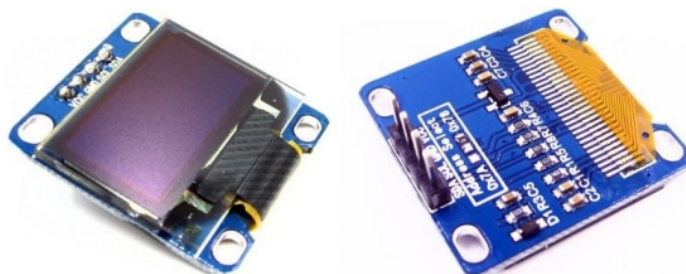
1x Computer with Arduino IDE Software
1x USB 2.0 Type A/B Data Cable
1x Arduino Uno Board
1x i2c SSD1306 OLED Module 64x128pixel

OLED GND to Arduino Uno GND
OLED VCC to Arduino Uno 3.3V
OLED SDA to Arduino Uno Pin 8
OLED SCL to Arduino Uno Pin 9

Note: Sometimes SCL is also labelled
as SCK

There are alot of codes in this Program. I wont post the full source codes
here. You can download the full Source codes from my github page.

https://github.com/teaksoon/lmaewapm/blob/main/
2021_11_22_io_pin_output_oled_source.zip

After download the Source Code, open the .zip file into a folder. Open the
Source Code with your Arduino IDE Software and upload it into your ATMEGA328
micro-controller on your Arduino Uno.

You will see some activities on your i2c SSD1306 OLED Screen

This is just a regular OUTPUT I/O Pin VOLTAGE manipulation with 0V and 5V at
a specific sequence, we can get things shown on the OLED Screen.

Earlier we only see the LED being turned ON and OFF with our digitalWrite(),
now we are just being **a little bit more creative and with the 0V and 5V than
previously, and we get something displayed on the OLED Screen.**

**Actually it is not just the LED or OLED Screen only, there are
alot more to it**

Besides the "digitalWrite()" of the "C-Language with Arduino Library"
included is also the codes for the "C-Language with AVR Library" which can
run much faster, which will be very useful if we were to run alot of
graphics on the OLED Screen.

This example is split into 3 files.
1. io_output_oled.ino is the test Program
2. io_oled_arduino.h is coding using the "C-Language with Arduino Library"
3. io_oled_avr.h is coding using the "C-Language with AVR Library"

This code in io_output_oled.ino uses codes from the io_oled_arduino.h
The top part must be commeented off

```
/*
#include "io_oled_avr.h"
// Default Pin for SDA=PB0,SCL=PB1, change below if other pins are used
#define PIN_SDA PB0  // ( which is Arduino Uno Pin 8 )
#define PIN_SCL PB1  // ( which is Arduino Uno Pin 9 )
*/

#include "io_oled_arduino.h"
// Default Pin for SDA=8,SCL=9, change below if other pins are used
#define PIN_SDA 8
#define PIN_SCL 9
```

This code in io_output_oled.ino uses codes from the io_oled_avr.h
The bottom group must be commented off

```
#include "io_oled_avr.h"
// Default Pin for SDA=PB0,SCL=PB1, change below if other pins are used
#define PIN_SDA PB0  // ( which is Arduino Uno Pin 8 )
#define PIN_SCL PB1  // ( which is Arduino Uno Pin 9 )

/*
#include "io_oled_arduino.h"
// Default Pin for SDA=8,SCL=9, change below if other pins are used
#define PIN_SDA 8
#define PIN_SCL 9
*/
```

If you wish to use other I/O Pins, just change the #define PIN_SDA and
#define PIN_SCL to other valid Pin Numbers

Source Codes responsible for the 0V/5V manipulation on the OUTPUT I/O Pin to
make the OLED Screen show things, using digitalWrite();
from the io_oled_arduino.h file

```c
void _io_oled_i2c_start() {
  digitalWrite(_io_pinSDA, HIGH); __asm__("NOP");
  digitalWrite(_io_pinSCL, HIGH); __asm__("NOP");
  digitalWrite(_io_pinSDA,  LOW); __asm__("NOP");
  digitalWrite(_io_pinSCL,  LOW); __asm__("NOP");
}

void _io_oled_i2c_stop() {
  digitalWrite(_io_pinSDA,  LOW); __asm__("NOP");
  digitalWrite(_io_pinSCL, HIGH); __asm__("NOP");
  digitalWrite(_io_pinSDA,  LOW); __asm__("NOP");
}

uint8_t io_oled_i2c_write_byte(uint8_t dat) {
uint8_t ack=0;  // 0=successful, 1=failed
  for(uint8_t i=0; i < 8; i++) {
    if (dat&0x80) {
      digitalWrite(_io_pinSDA, HIGH);
    } else {
      digitalWrite(_io_pinSDA, LOW);;
    }
    dat<<=1;
     __asm__("NOP");
    digitalWrite(_io_pinSCL, HIGH);  __asm__("NOP");
    digitalWrite(_io_pinSCL,  LOW);  __asm__("NOP");
  }
  digitalWrite(_io_pinSDA, HIGH);
  digitalWrite(_io_pinSCL, HIGH); __asm__("NOP");
  digitalWrite(_io_pinSCL,  LOW);
  return ack;
}
```

I am sure everyone here already know what digitalWrite() does to the OUTPUT
I/O Pin.

_io_pinSDA and **_io_PinSCL** are basically our regular Pin Number

Maybe **__asm__** is an alien to some of you. That is an **inline Assembly Code,
calling a single INSTRUCTION from our AVR INSTRUCTION SET.**

"NOP" simply means **"NO OPERATIONS",** it is spending 1-clock cycle doing
nothing. (bascially to slow down things. Actually not necessary when using
digitalWrite(), because digitalWrite() function is slow enough).

The reason I kept them there is for use with the "C-Language with AVR
Libraries" CPU Memory PORT manipulation, which is just 2-cycle for each
OUTPUT I/O Pin Update, so the "NOP" delay is necessary to slow down things

Source Codes responsible for the 0V/5V manipulation on the OUTPUT I/O Pin to make the OLED Screen show things, using direct CPU Memory PORT manipulation from the io_oled_avr.h file

```c
void _io_oled_i2c_start() {
   (PORTB |= (1<< _io_pinSDA)); __asm__("NOP");
   (PORTB |= (1<< _io_pinSCL)); __asm__("NOP");
   (PORTB &= ~(1<<_io_pinSDA)); __asm__("NOP");
   (PORTB &= ~(1<<_io_pinSCL)); __asm__("NOP");
}

void _io_oled_i2c_stop() {
   (PORTB &= ~(1<<_io_pinSDA)); __asm__("NOP");
   (PORTB |= (1<< _io_pinSCL)); __asm__("NOP");
   (PORTB |= (1<< _io_pinSDA)); __asm__("NOP");
}

uint8_t io_oled_i2c_write_byte(uint8_t dat) {
uint8_t ack = 0;  // 0=successful, 1=failed
   for(uint8_t i=0; i < 8; i++) {
      (dat & 0x80) ? (PORTB |= (1<< _io_pinSDA)) : (PORTB &= ~(1<<
_io_pinSDA));
      dat<<=1;
       __asm__("NOP");

      (PORTB |= (1<< _io_pinSCL));  __asm__("NOP");
      (PORTB &= ~(1<<_io_pinSCL));  __asm__("NOP");
   }
   (PORTB |= (1<<_io_pinSDA));
   (PORTB |= (1<<_io_pinSCL)); __asm__("NOP");
   (PORTB &= ~(1<<_io_pinSCL));
   return ack;
}
```

IF you are confused with the codes, means we need to build a stronger foundation before can achieve something big, othwerwise after building that that "Robotic SmartCar", thats the end of our journey.

                    We can do far more than that