Digital I/O Operations

CSE0420 – Embedded Systems

By

Z. Cihan TAYŞİ

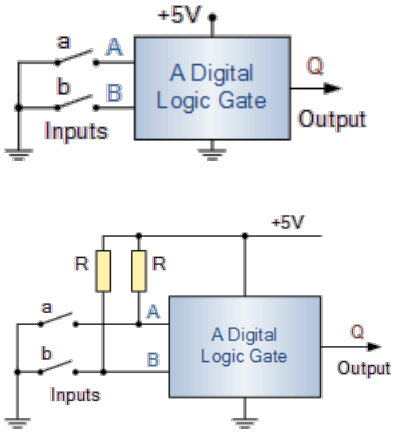
Outline

- Digital I/O
- Ports, Pins
- Direction
- Pull-up & pull-down
- Arduino programming
- Digital I/O examples on Arduino

Digital I/O

- Unlike analog signals, which may take on any value within a range of values, digital signals have two distinct values:
  - HIGH (1) or LOW (0).
- You use digital signals in situations, where the input or output will have one of those two values.
  - For example, one way that you might use a digital signal is to turn an LED on or off.
- Digital I/O pins can be used for either input or output
  - so you should set the direction of i/o pings

Digital I/O – (Pull-up & Pull-down)



- The two switches, “a” and “b”, represent the inputs to a generic logic gate.
  - When switch “a” is closed (ON), input “A” is connected to ground, (0v) or logic level “0” (LOW) and likewise,
  - However, when switch “a” is opened (OFF), what will be the value of the voltage applied to input “A”, HIGH or LOW?
- We assume it will be +5V (HIGH) as switch “a” is open-circuited and therefore input “A” is not shorted to ground, but this may not be the case. As the input is now effectively unconnected from either a defined HIGH or LOW condition, it has the potential to “float” about between 0V and +5V (Vcc)
- This uncertain situation may cause the digital input at “A” to stay at a logic level “0” (LOW) when the switch is open, when we actually need a logic “1”, (HIGH) causing the logic gate to falsely switch the output at “Q”.
  - Also once there, this floating and weak input signal could easily change value at the slightest of interference or noise from its neighbouring inputs or could even cause it to go into oscillation, rendering the gate practically unusable.

Aurduino Programming – I

- The basic structure of the Arduino programming language is fairly simple and runs in at least two parts
  - void setup()
  - void loop()
- setup() is the preparation, loop() is the execution.
- Both functions are required for program to work.

Aurduino Programming – II

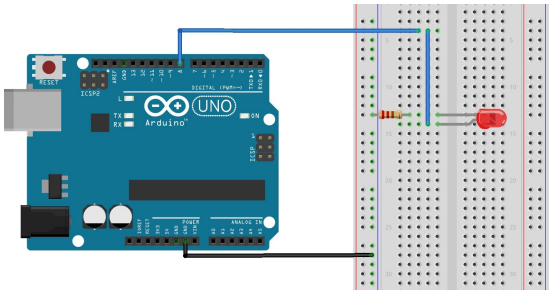
- The **setup()** function called once, when your program starts. It must be included in the program even if there are no statements to run. Use it to
  - initialize pin modes
  - setup serial connection
- After calling the setup() function, the **loop()** function does what its name suggests, and loops consectively.

Digital I/O on Arduino

- pinMode(pin, direction);
  - used to configure a specified pin to behave either as an INPUT or OUTPUT.
- digitalRead(pin);
  - reads the value from a specified digital pin with result either HIGH or LOW.
- digitialWrite(pin, value);
  - outputs either logic level HIGH or LOW at a specified digital pin.
- delay();
  - pauses your program for the among of time as specified in miliseconds, where 1000 equals second
- milis();
  - returns the number of miliseconds since the Arduino board began running the current program as an unsigned long value

Arduino Example #1

- **Blinking LED**
- **Hardware Required**
  - 1 x LED
  - 1 x 220 ohm resistor
  - 1 x Arduino UNO
  - 1 x breadboard
  - 2 x jumper Wires



Arduino Example #1

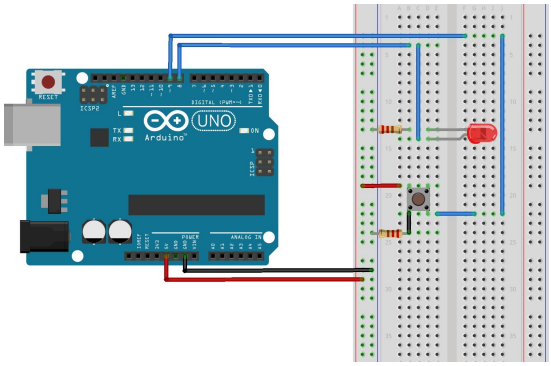
```
const int led = 8; //use digital I/O pin 8

void setup() {
     pinMode(led,OUTPUT);   // set pin 8 to be an output
}

void loop() {
     delay(1000); //delay 1000 milliseconds
     digitalWrite(led,HIGH); // turning on LED
     delay(1000);             // delay 1000 milliseconds
     digitalWrite(led,LOW);  // turning off LED
}
```

Arduino Example #2

- **Turn on/off LED**
- **Hardware Required**
  - 1 x LED
  - 2 x 220 ohm resistor
  - 1x pushbutton switch
  - 1 x Arduino UNO
  - 1 x breadboard
  - 6 x jumper Wires



Arduino Example #2

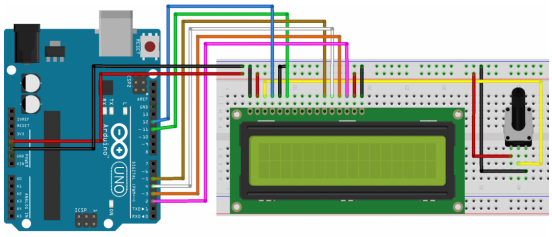
```
const int led = 8;           //name pin 8 as led
const int button = 9;        //name pin 9 as button
void setup() {
     pinMode(led,OUTPUT);    //set pin 8 as OUTPUT
     pinMode(button,INPUT) ; //set pin 9 as INPUT
}
void loop() {
     int reads = digitalRead(button); //read the digital
value on pin 9
     digitalWrite(led,reads); //set the digital output
value of pin 8 to that value
}
```

Arduino Example #3

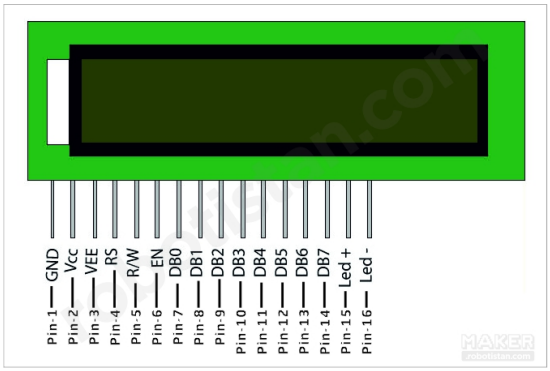
- Alter last example
- Write an application that changes the state (on/off) of the led, when the button is pressed
  - initial status, the led is off
  - when the button is pressed, the led turns on
  - when the button is pressed again, the led turns off
  - if the button pressed again, the led changes its state again
- Pin 13 is already connected to onboard LED
- Pin 2 is used for button

Arduino Example #4

- **Interfacing a 2x16 LCD**
- **Hardware Required**
  - 1 x Arduino UNO
  - 1 x breadboard
  - 16 x 2 LCD display
  - 5k potentiometer
  - Several jumper cables

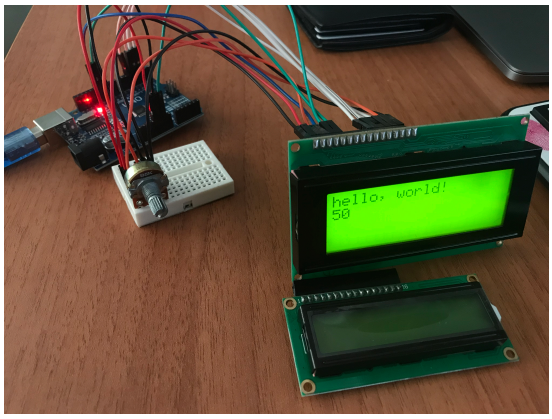


Arduino Example #4



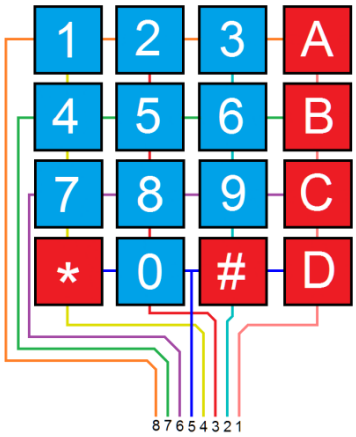
- **Data Bus:** As shown in the figure, an alphanumeric LCD has an 8-bit data bus referenced as D0-D7. As it is an 8-bit data bus, we can send the data/cmd to LCD in bytes. It also provides the provision to send the data/cmd in chunks of 4-bit, which is used when there are limited number of GPIO lines on the microcontroller.
- **Register Select(RS):** The LCD has two register namely a Data register and Command register. Any data that needs to be displayed on the LCD has to be written to the data register of LCD. Command can be issued to LCD by writing it to Command register of LCD.
- **Read/Write(RW):** This signal is used to write the data/cmd to LCD and reads the busy flag of LCD.
  - For write operation the RW should be **LOW**
  - and for read operation the R/W should be **HIGH**.
- **Enable(EN):** This pin is used to send the enable trigger to LCD.

Arduino Example #4



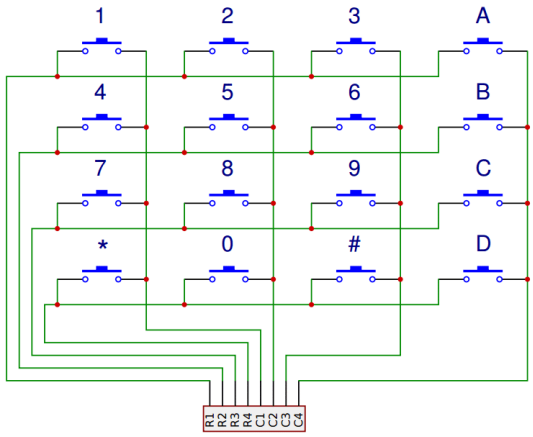
Arduino Example #5

- **A Simple Keyboard**
- **Hardware Required**
  - 1 x Arduino UNO
  - 1 x breadboard
  - 4 buttons
  - Several jumper cables
  - **Serial connection**

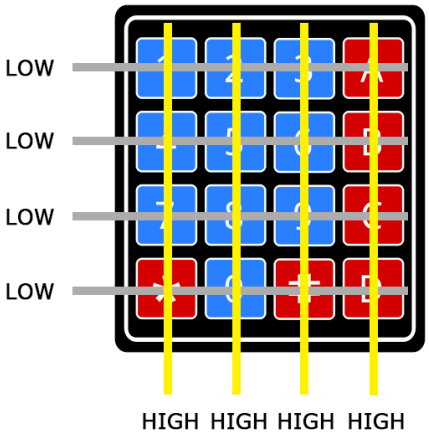


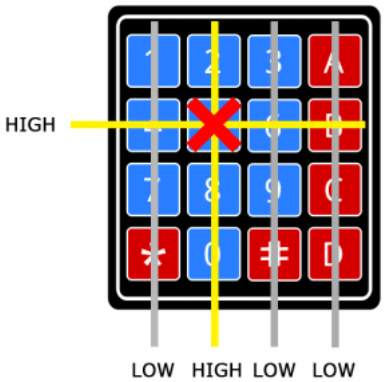
Arduino Example #5

- Matrix keypads use a combination of rows and columns to provide button states to the microcontroller.
  - Underneath each key is a pushbutton, with one end connected to one row, and the other end connected to one column.
  - In order for the microcontroller to determine which button is pressed, it first needs to pull each of the four columns (pins 1-4) either low or high one at a time,
  - and then poll the states of the four rows (pins 5-8).
  - Depending on the states of the columns, the microcontroller can tell which button is pressed.



Arduino Example #5





References

- https://www.allaboutcircuits.com/projects/learn-how-to-use-the-arduinos-digital-i-o/
- https://playground.arduino.cc/uploads/Main/arduino_notebook_v1-1.pdf
- https://maker.robotistan.com/arduino-dersleri-10-16x2-lcd-ekran/
- http://osoyoo.com/2017/09/13/arduino-lesson-4x4-matrix-keypad/