I/O Programming and Interfacing

Prof. Dhaval Shah

Outline

- I/O interface
- Digital IO functions
- LED interfacing
- Push button interfacing
- Button Debounce
- Hex Keypad interfacing
- Seven segment interfacing

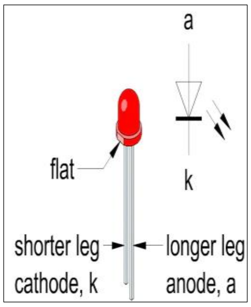
I/O interfacing

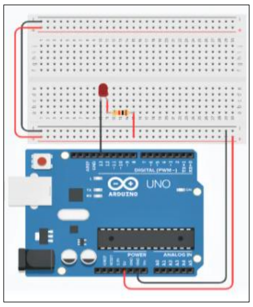
- An **input/output** pin, or I/O pin, is the interface between a microcontroller and another circuit.
- Used to exchange the information between two devices
- The digital pins on an Arduino board can be used for general purpose input and output via the pinMode(), digitalRead(), and digitalWrite() commands.

Digital IO functions

1. pinMode() : Configures the specified pin to behave either as an input or an output
   - Syntax : pinMode(pin, mode)
     - Pin: the Arduino pin number to set the mode of
     - Mode: INPUT, OUTPUT or INPUT_PULLUP
2. digitalRead() : Read the value from a specified digital pin, either HIGH or LOW
   - Syntax : digitalRead(pin, value)
3. digitalWrite () : Write a HIGH or LOW to a digital pin
   - Syntax : digitalRead(pin, value)

LED interfacing





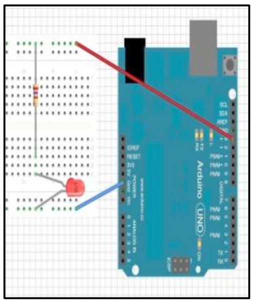
How to write a code in Arduino?

- Define led pin--→ use constant int type

  int led = 13;

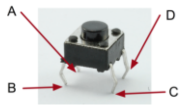
- Setup the pin modes for led under void setup
  - pinMode(ledPin, OUTPUT);
  - pinMode(buttonpin, INPUT);
- Define the on/off conditions under void loop

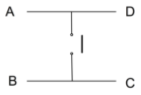
```
void loop() {
digitalWrite(led, HIGH); // turn the LED on

delay(1000); // wait for a second
digitalWrite(led,  LOW);  //  turn  the  LED  off

delay(1000); // wait for a second
}
```



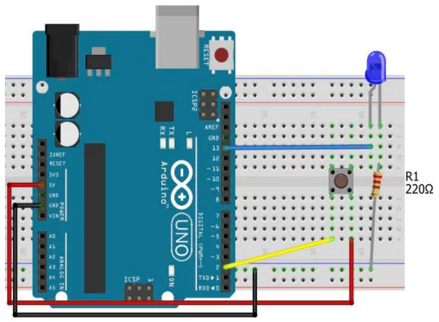
Push Button Switch

- When you press a button or flip a lever, they connect two contacts together so that current flow through them.
- There are only really two electrical connections, as inside the switch package pins B and C are connected together, as are A and D.





Push Button Switch - Interfacing



Push Button Switch - Interfacing

- Define button pin --→ use a constant int type

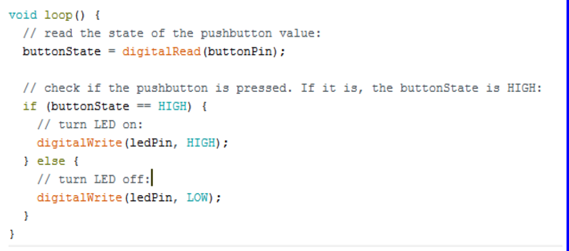
  const int buttonPin = 2;
- Define led pin--→ use constant int type

  const int ledPin = 10;
- Initialize button state -→

  int buttonState = 0;
- Setup the pin modes for both the pins under void setup
  – pinMode(ledPin, OUTPUT);
  – pinMode(buttonpin, INPUT);

Push Button Switch - Interfacing

- Set the conditions for LED on or off under void loop after reading button state



```
void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

Push Button Switch - Debounce

- System that counts the **number of times a button** is pressed, One may count individual presses as multiple hits. The solution to this problem is called denouncing
- pushbutton or **any switch's position is changed noise is generated**.
  – noise (contact) occurs because the switch contact is metal and it has elasticity
- contact bounce: switch is moved to a new position it strikes a metal contact and physically bounces a few times.
- Bouncing happens in a matter of milliseconds – but microcontroller is moving so fast that it will detect a transition between two states every time the button bounces

Push Button Switch - Debounce

- // Define the pins (constants won't change)

  const int buttonPin = 2; // the number of the pushbutton pin

  const int ledPin = 13; // the number of the LED pin
- // Variables will change:

  int ledState = HIGH; // the current state of the output pin

  int buttonState; // the current reading from the input pin

  int lastButtonState = LOW; // the previous reading from the input pin
- // New variables are unsigned longs because the time, measured in milliseconds, will quickly become a bigger number than can be stored in an int.

  unsigned long lastDebounceTime = 0; // the last time the output pin was toggled

  unsigned long debounceDelay = 50; // the debounce time; increase if the output flickers

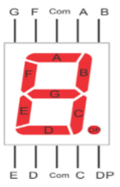
Cont…

```
• // Define pin modes in setup loop
    void setup() {
      pinMode(buttonPin, INPUT);
      pinMode(ledPin, OUTPUT);
      // set initial LED state
      digitalWrite(ledPin, ledState);
    }
• //Define operation in loop
void loop() {
int reading = digitalRead(buttonPin); // read the state of the switch into a local variable
• // check to see if you just pressed the button i.e. the input went from LOW to HIGH),
  and you've waited long enough since the last press to ignore any noise:
• // If the switch changed, due to noise or pressing:
      if (reading != lastButtonState) {
    lastDebounceTime = millis(); // reset the debouncing timer
      }
```

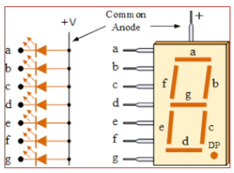
Cont…

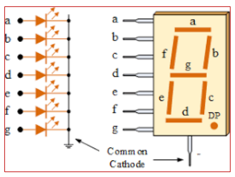
```
        if ((millis() - lastDebounceTime) > debounceDelay) {
          // whatever the reading is at, it's been there for longer than the debounce delay,
        so take it as the actual current state:
          // if the button state has changed:
          if (reading != buttonState) {
            buttonState = reading;
            // only toggle the LED if the new button state is HIGH
            if (buttonState == HIGH) {
              ledState = !ledState;
            }
          }
        }
digitalWrite(ledPin, ledState); // set the LED:
  // save the reading. Next time through the loop, it'll be the lastButtonState:
  lastButtonState = reading;
}
```

Seven Segment Display

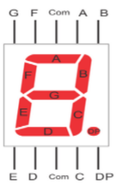


- Two types
  - Common Anode
  - Common Cathode





Mapping Table



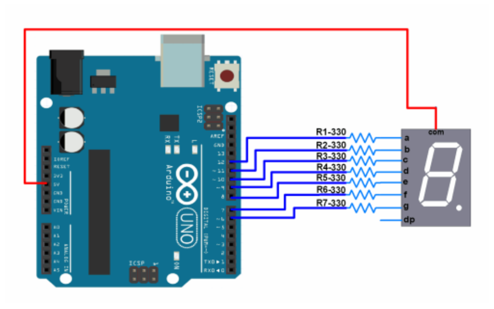
Common Cathode

Digital	a	b	c	d	e	f	g
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0

Common Anode

Digital	a	b	c	d	e	f	g
0	0	0	0	0	0	0	1
1	1	0	0	1	1	1	1

Interfacing with Arduino



Interfacing with Arduino

- Write a code to display number in order or
- turn LEDs on and off in order to become familiar with how a seven-segment display functions.
  - Declare pins
  - Setup mode of each pin

```
pinMode(2,OUTPUT);
pinMode(3,OUTPUT);
pinMode(4,OUTPUT);
pinMode(5,OUTPUT);
pinMode(6,OUTPUT);
pinMode(7,OUTPUT);
 pinMode (8,OUTPUT);
```

Cont…

```
void loop()
{
  // loop to turn leds od seven seg ON

  for(int i=2;i<9;i  )
  {
    digitalWrite(i,HIGH);
    delay(600);
  }

  // loop to turn leds od seven seg OFF
  for(int i=2;i<9;i  )
  {
    digitalWrite(i,LOW);
    delay(600);
  }


  delay(1000);

}
```

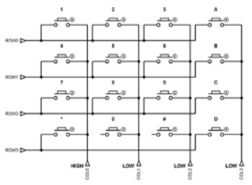
```
  int disp_pin[7]; /* array for a-g pins of 7-Segment display */

  void define_segment_pins(int a, int b, int c, int d, int e, int f, int g)
  {
    disp_pin[0] = a;
    disp_pin[1] = b;
    disp_pin[2] = c;
    disp_pin[3] = d;
    disp_pin[4] = e;
    disp_pin[5] = f;
    disp_pin[6] = g;
  }

void display_number(int num)      /* Function for displaying number (0-9) */
{
  switch(num)
  {
    case 0:
    digitalWrite(disp_pin[0], LOW);      /* Drive disp_pin[0] to LOW */
    digitalWrite(disp_pin[1], LOW);      /* Driving LOW turns on LED segment
    digitalWrite(disp_pin[2], LOW);
    digitalWrite(disp_pin[3], LOW);
    digitalWrite(disp_pin[4], LOW);
    digitalWrite(disp_pin[5], LOW);
    digitalWrite(disp_pin[6], HIGH);
    break;
```

HEX Keypad

- It is simply an arrangement of 16 push button switches in a 4X4 matrix form.
- Typical applications are : code locks, calculators, automation systems or simply any thing that requires a character or numeric input

HEX Keypad – Column Scanning

- The most common way to implement this by making a column pin high while making the rest of the columns low and do that in sequence
- column "0" is high and the rest of the columns are low. If you press the "1" button, row "0" will be high because by then it will be connected to column "0".
- scan row "0" using *digitalWrite()* while column "0" is high, this means the user pressed button "1".

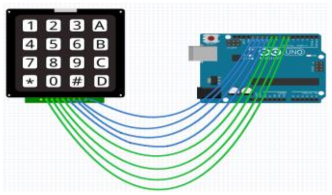


Cont…

When this button is pressed...	If...	Then...
1	Col0=high	Row0=high
2	Col1=high	Row0=high
3	Col2=high	Row0=high
4	Col0=high	Row1=high
5	Col1=high	Row1=high
6	Col2=high	Row1=high
7	Col0=high	Row2=high
8	Col1=high	Row2=high
9	Col2=high	Row2=high
0	Col1=high	Row3=high
*	Col0=high	Row3=high
#	Col2=high	Row3=high
A	Col3=high	Row0=high
B	Col3=high	Row1=high
C	Col3=high	Row2=high
D	Col3=high	Row3=high

HEX Keypad Interfacing

- The hex keypad will have 8 connection wires namely R1, R2, R3, R4 and C1, C2, C3, C4 representing the rows and columns respectively.



HEX Keypad Interfacing - Code

- Define the Rows and Column

```
int ROWS[4] = {10,11,12,13};
int COLS[4] = {6,7,8,9};
```

- Define row pins as input and column as a output in void setup

```
void setup() {
pinMode(6, OUTPUT);   pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);   pinMode(9, OUTPUT);
  pinMode(10, INPUT);     pinMode(11, INPUT);
  pinMode(12, INPUT);    pinMode(13, INPUT);
```

- Set a serial monitor to check the output within void setup

```
 Serial.begin(9600); //For printing out the output
 }
```

- void loop() {
- //Defining a column state

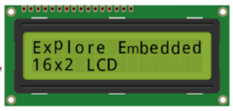
```
 digitalWrite(COLS[0],HIGH);   digitalWrite(COLS[1],LOW);
digitalWrite(COLS[2],LOW);   digitalWrite(COLS[3],LOW);
```

Cont…

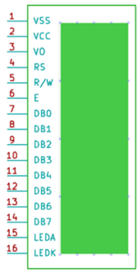
```
  if(digitalRead(ROWS[0]) == HIGH && digitalRead(ROWS[1]) == LOW &&
digitalRead(ROWS[2]) == LOW && digitalRead(ROWS[3]) == LOW)
               {  Serial.println("1"); }
 else if(digitalRead(ROWS[0]) == LOW && digitalRead(ROWS[1]) == HIGH
&& digitalRead(ROWS[2]) == LOW && digitalRead(ROWS[3]) == LOW)
       {Serial.println("4");  }
else if(digitalRead(ROWS[0]) == LOW && digitalRead(ROWS[1]) == LOW
&& digitalRead(ROWS[2]) == HIGH && digitalRead(ROWS[3]) == LOW)
      { Serial.println("7"); }
else if(digitalRead(ROWS[0]) == LOW && digitalRead(ROWS[1]) == LOW
&& digitalRead(ROWS[2]) == LOW && digitalRead(ROWS[3]) == HIGH)
      { Serial.println("*"); }
else{;}
 delay(100);
```

Repeat the similar way by changing the column state for other column

LCD Interfacing

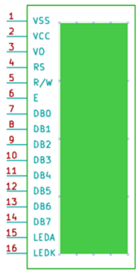


Pin Number	Symbol	Pin Function
1	VSS	Ground
2	VCC	+5v
3	VEE	Contrast adjustment (VO)
4	RS	Register Select. 0:Command, 1: Data
5	R/W	Read/Write, R/W=0: Write & R/W=1: Read
6	EN	Enable. Falling edge triggered
7	D0	Data Bit 0 (Not used in 4-bit operation)

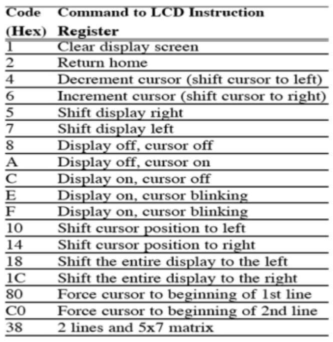


LCD Interfacing

Pin Number	Symbol	Pin Function
8	D1	Data Bit 1 (Not used in 4-bit operation)
9	D2	Data Bit 2 (Not used in 4-bit operation)
10	D3	Data Bit 3 (Not used in 4-bit operation)
11	D4	Data Bit 4
12	D5	Data Bit 5
13	D6	Data Bit 6
14	D7	Data Bit 7/Busy Flag
15	A/LED+	Back-light Anode(+)
16	K/LED-	Back-Light Cathode(-)



LCD Commands



Code (Hex)	Command to LCD Instruction Register
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning of 1st line
C0	Force cursor to beginning of 2nd line
38	2 lines and 5x7 matrix

LCD functions in Arduino

- LiquidCrystal()
  - Creates a variable of type LiquidCrystal.
  - The display can be controlled using 4 or 8 data lines.
  - Syntax
    - LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)
    - LiquidCrystal(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)
  - Example
    - LiquidCrystal lcd(12, 11, 10, 5, 4, 3, 2);

Cont…

- begin()
  - Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display
  - Syntax
    - lcd.begin(cols, rows)
  - Example
    - lcd.begin(16, 2);
- clear()
  - Clears the LCD screen and positions the cursor in the upper-left corner
  - Syntax
    - lcd.clear();

Cont…

- home()
  - Positions the cursor in the upper-left of the LCD.
  - Can use lcd.clrear(), as it does same function
- setCursor()
  - Position the LCD cursor; that is, set the location at which subsequent text written to the LCD will be displayed.
  - Syntax
    - lcd.setCursor(col, row)
  - Example
    - lcd.setCursor(0, 1)

Cont…

- write()
  - Write a character to the LCD.
- print()
  - Prints text to the LCD.
- cursor()
  - Display the LCD cursor: an underscore (line) at the position to which the next character will be written.
- noCursor()
  - Hides the LCD cursor.
- blink()
  - Display the blinking LCD cursor.

Cont…

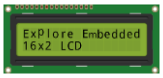
- display()
  - Turns on the LCD display, after it's been turned off with noDisplay(). This will restore the text (and cursor) that was on the display.
- scrollDisplayLeft()/scrollDisplayRight()
  - Scrolls the contents of the display (text and cursor) one space to the left/right.
- autoscroll()
  - Turns on automatic scrolling of the LCD. This causes each character output to the display to push previous characters over by one space.
- leftToRight()/rightToLeft()
  - Set the direction for text written to the LCD to left-to-right/ right-to-left

Steps to interface LCD with Arduino

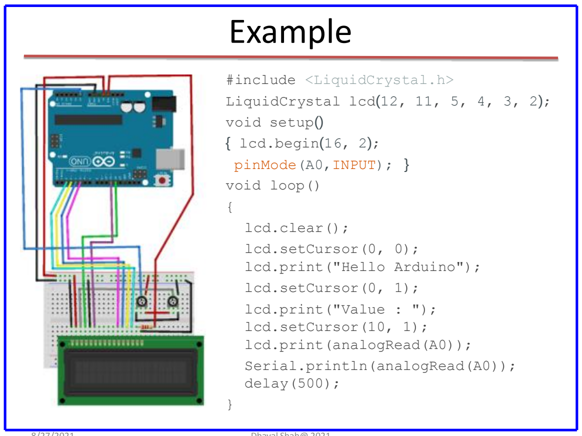
- Void Setup part
- Step: 1
  - Include the library LiquidCrystal.h
  - Syntax: #include <LiquidCrystal.h>
- Step: 2
  - Initialize the library with the number of the interface pins. With the function "LiquidCrystal lcd()".
  - Syntax: LiquidCrystal lcd(RS, E, D4, D5, D6, D7);
  - LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
- Step: 3
  - Set the number of columns and rows by the function "lcd.begin(16, 2)“
  - Define other pins, if any

Cont…

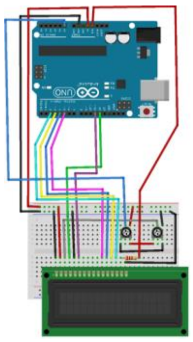
- Void loop part
- Step: 4
  - clear the display using the function "lcd.clear()"
- Step: 5
  - set the cursor to that particular point (starting point) by the function "lcd.setCursor()
  - Syntax: lcd.setCursor(0, 0);



- Step: 6
  - Print the character/data by the function "lcd.print(“Hello Arduino”);
  - This will print on first row as per the current setting



Example



```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup()
{ lcd.begin(16, 2);
 pinMode(A0,INPUT); }
void loop()
{
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Hello Arduino");
  lcd.setCursor(0, 1);
  lcd.print("Value : ");
  lcd.setCursor(10, 1);
  lcd.print(analogRead(A0));
  Serial.println(analogRead(A0));
  delay(500);
}
```