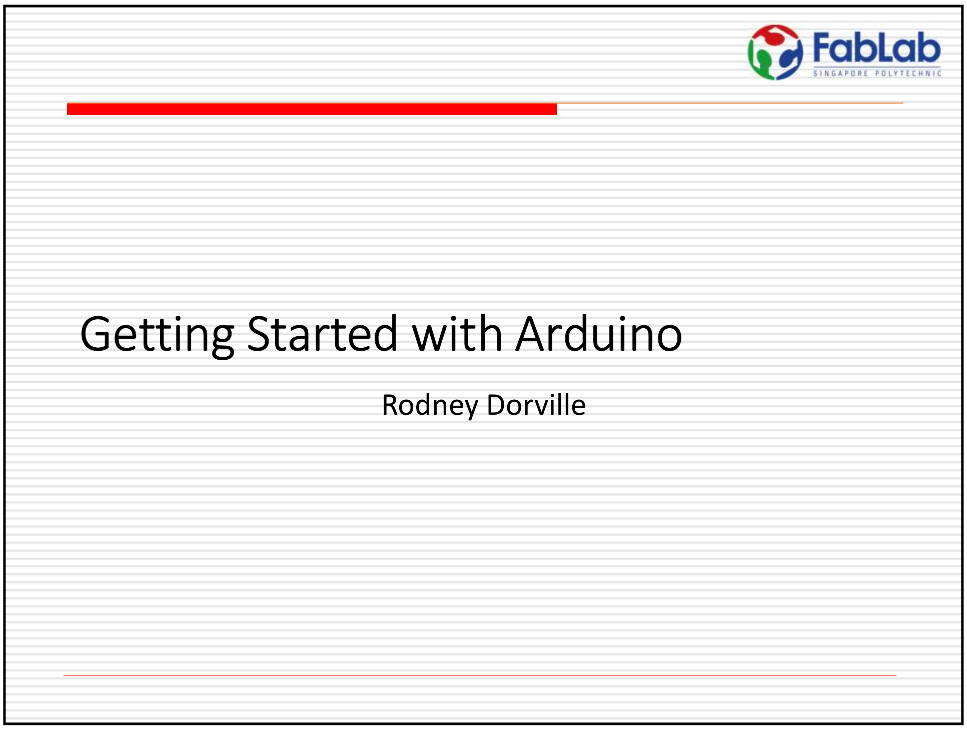
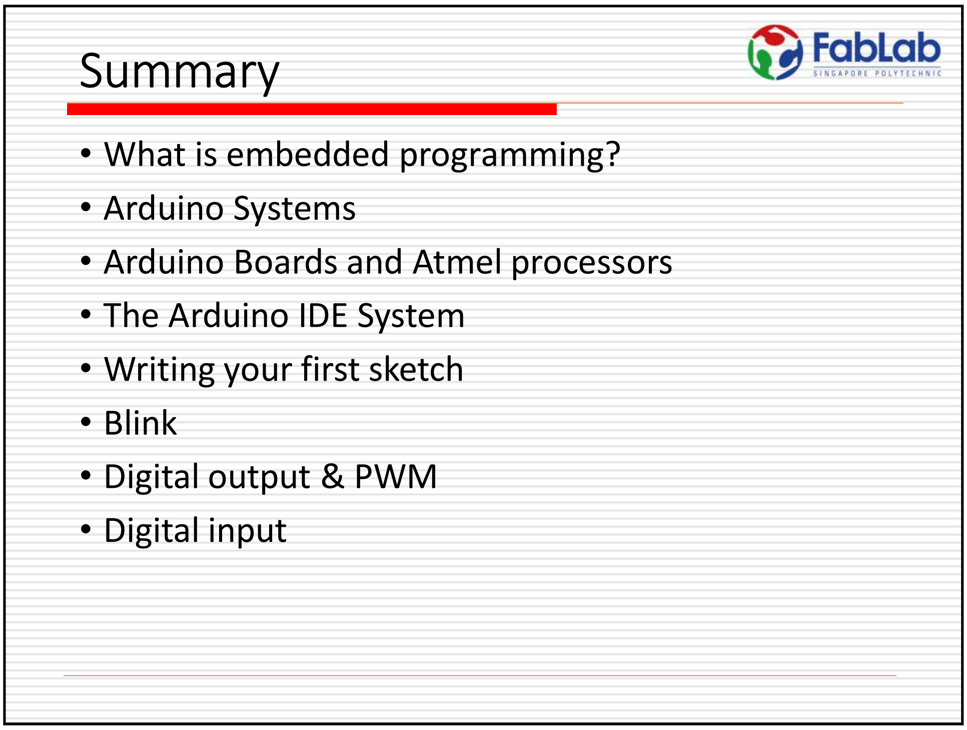


Getting Started with Arduino

Rodney Dorville

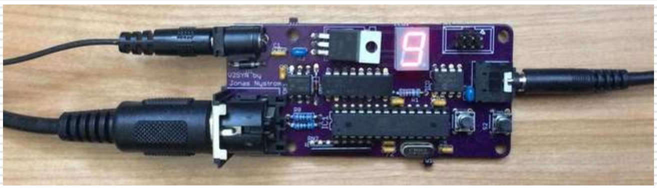


Summary

- What is embedded programming?
- Arduino Systems
- Arduino Boards and Atmel processors
- The Arduino IDE System
- Writing your first sketch
- Blink
- Digital output & PWM
- Digital input

Embedded Programming

- A dedicated computer system with a dedicated function within a larger system to perform a specific task. E.g. smart TVs, ovens
- After development with the Uno, the circuit could be reduced to a single IC with support devices.



Ref: MatrixSynth – ATMega328 synthesizer

What is an Arduino?

- An Arduino System comprises of
  - ❑ Software & Software Tools
    - ❖ Integrated Development System (IDE)
    - ❖ Arduino programming language (C++ like) based on Wiring
    - ❖ Development & Debugging tool based on Processing
    - ❖ Software libraries (open source contributions)
  - ❑ Hardware
    - ❖ ATMel processor board
    - ❖ Shields (Add-on modules)
    - ❖ Sensors, actuators, peripherals
- Open Source Platform

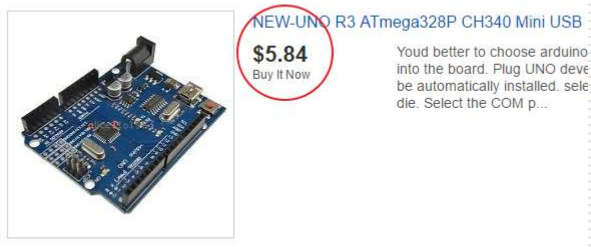
Ref: Open Source Software

Arduino Systems

- A hardware and software company based in Italy
- Initial development – software (based on Wiring) and later hardware boards (Atmel based)
- Produces and markets “official” boards: Uno, Due, Leonardo, Diecimila, Mega, Nano
- Software and hardware is open source.

Why use Arduino?

- Inexpensive (cost < $30)
- Cross-platform (Mac, Linux, Windows)
- Simple, clear programming environments using a GUI
- Open source



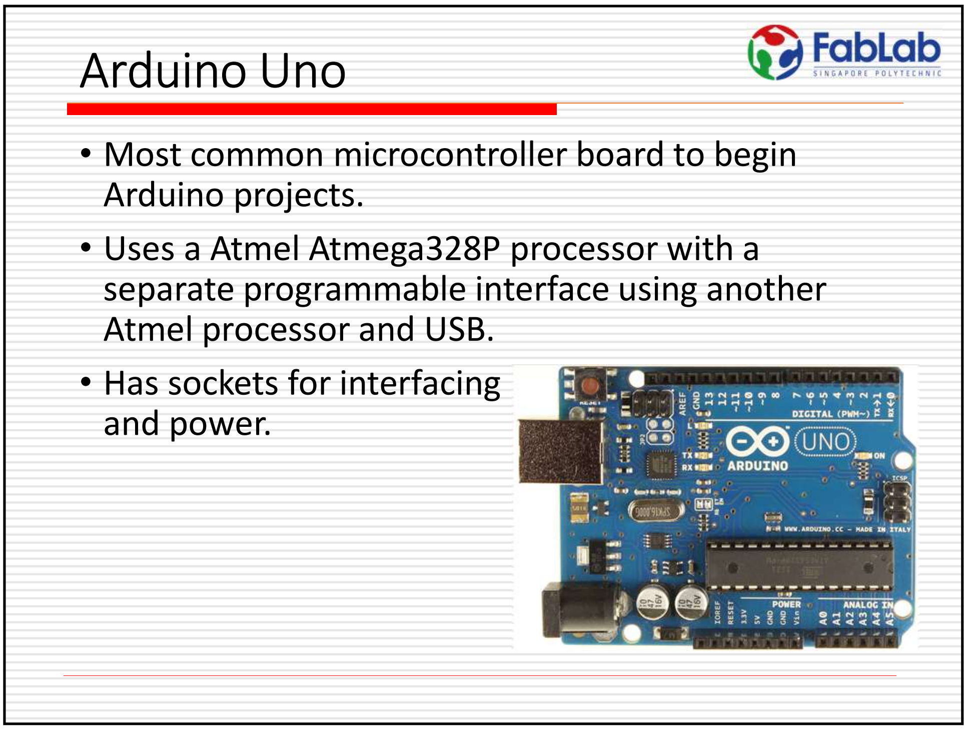
NEW-UNO R3 ATmega328P CH340 Mini USB

$5.84

Buy It Now

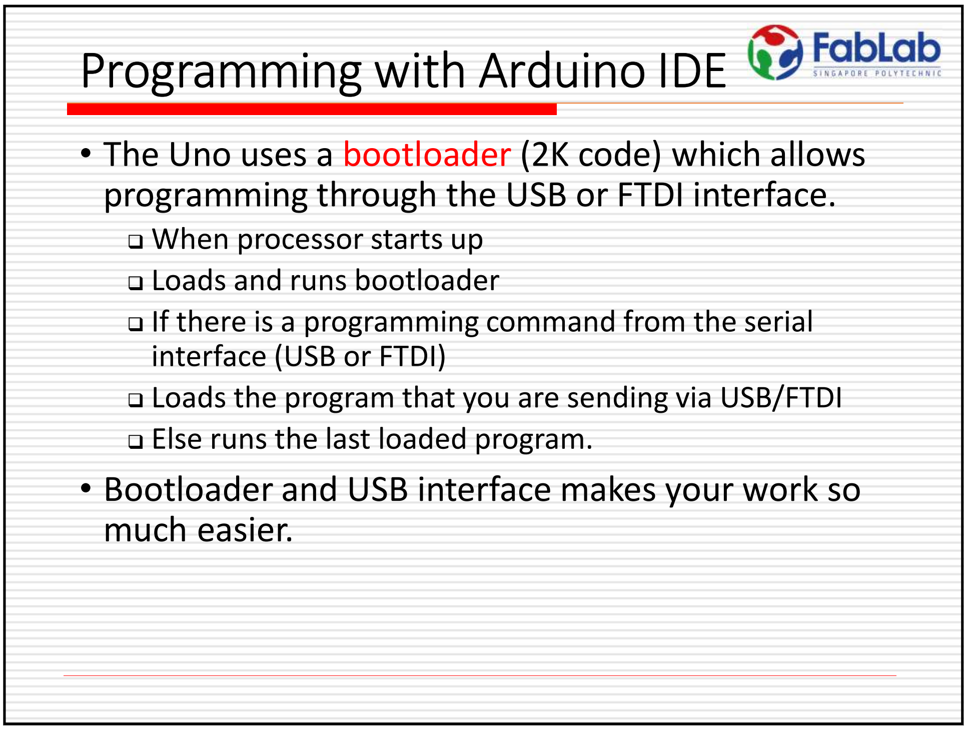
Youd better to choose arduino into the board. Plug UNO deve be automatically installed. sele die. Select the COM p...

Ref: Arduino – Why use Arduino?



Arduino Uno

- Most common microcontroller board to begin Arduino projects.
- Uses a Atmel Atmega328P processor with a separate programmable interface using another Atmel processor and USB.
- Has sockets for interfacing and power.

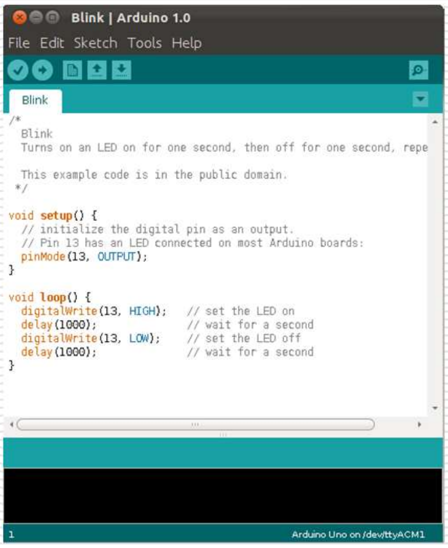


Programming with Arduino IDE

- The Uno uses a bootloader (2K code) which allows programming through the USB or FTDI interface.
  - ❑ When processor starts up
  - ❑ Loads and runs bootloader
  - ❑ If there is a programming command from the serial interface (USB or FTDI)
  - ❑ Loads the program that you are sending via USB/FTDI
  - ❑ Else runs the last loaded program.
- Bootloader and USB interface makes your work so much easier.

Using the Arduino IDE

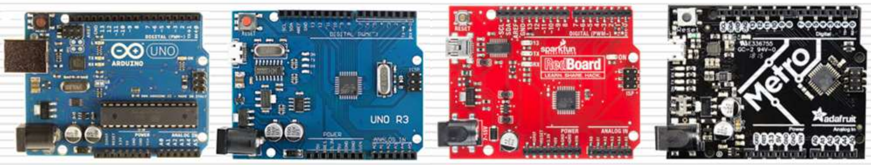
- Write your code
- Compile
- Upload to UNO board
- Press RESET button
- Observe results



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repe

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

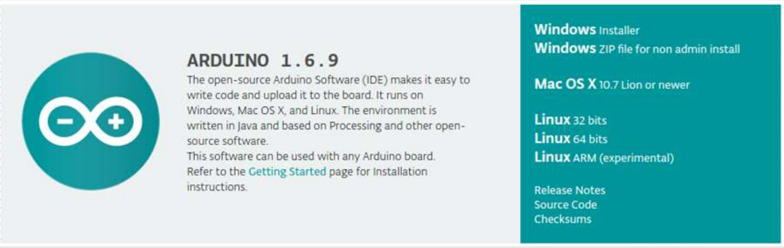
Variations of the UNO

- Being open-source, there are many variations.
- Programming and usage are basically the same with some minor variations.
- All boards use the ATMega328P processor (may be in different formats)
- All boards have the same I/O pins
- Difference is in $$cost$$



Arduino IDE Software

- Download and install the latest versions from the Arduino site.
- Current version 1.8.2
- Available in different platforms
- Copious help and how-tos available with simple search



ARDUINO 1.6.9

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board.

Refer to the Getting Started page for Installation instructions.

Windows Installer
Windows ZIP file for non admin install

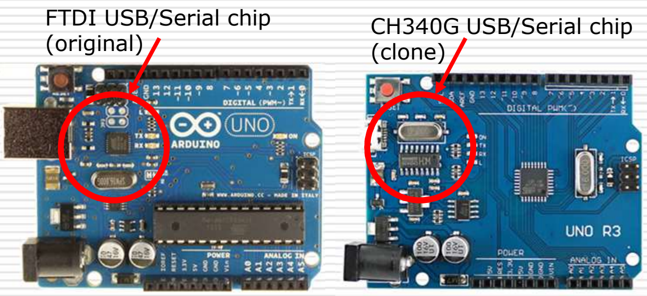
Mac OS X 10.7 Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM (experimental)

Release Notes
Source Code
Checksums

UNO board drivers

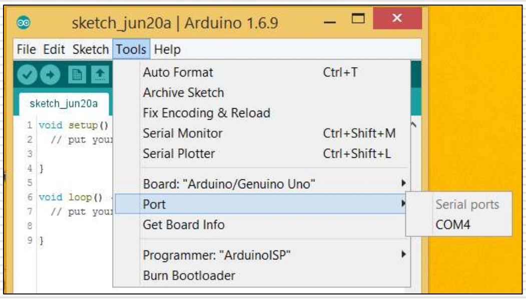
- Sometimes drivers to be installed.
  - ❑ Original Arduino boards use the FTDI drivers,
  - ❑ OEM boards use the cheaper CH340 drivers, which need to be installed.
- Plenty of help using Google


FTDI USB/Serial chip (original)
CH340G USB/Serial chip (clone)
ARDUINO
UNO
UNO R3

Ref: How to install Cheap China Arduinos that come with theCH340G/341G Serial/USB chip

Using the Arduino IDE

- You need to connect your UNO board to the host computer.
- Launch the Arduino IDE
- Setup the IDE
- Select the correct board that you are using (Tools>Board)

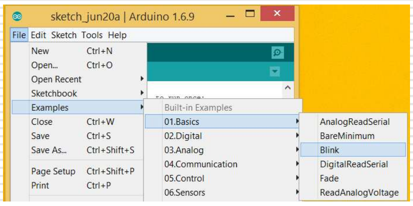


```
void setup()
  // put your
}

void loop()
  // put your
}
```

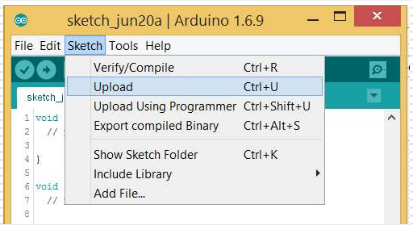
- Identify and check the port the board is connected to (Tools > Serial Port > (select the COM port))

Test a Sample program

- Load the example program “Blink”.
- Programs are called Sketches.



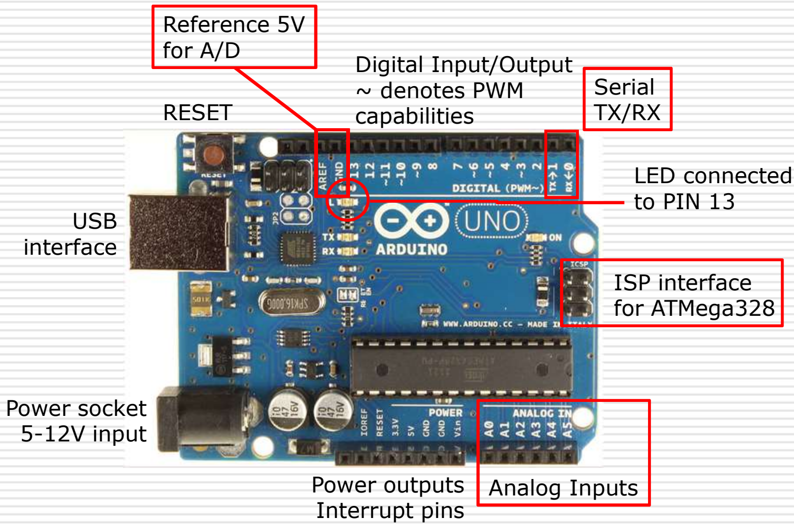
- Verify/Compile the program
- Upload


sketch_jun20a | Arduino 1.6.9
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
sketch_j
void
//
}
void
//

- Program executes after loading

Upload with programmer is only used with an ISP circuit

UNO board interfaces



Reference 5V for A/D

RESET

Digital Input/Output
~ denotes PWM
capabilities

Serial TX/RX

LED connected to PIN 13

USB interface

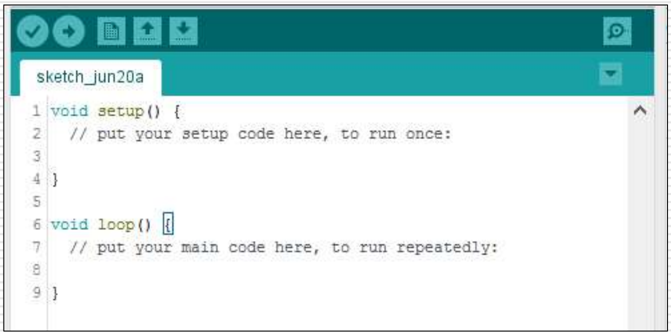
ISP interface for ATMega328

Power socket
5-12V input

Power outputs
Interrupt pins

Analog Inputs

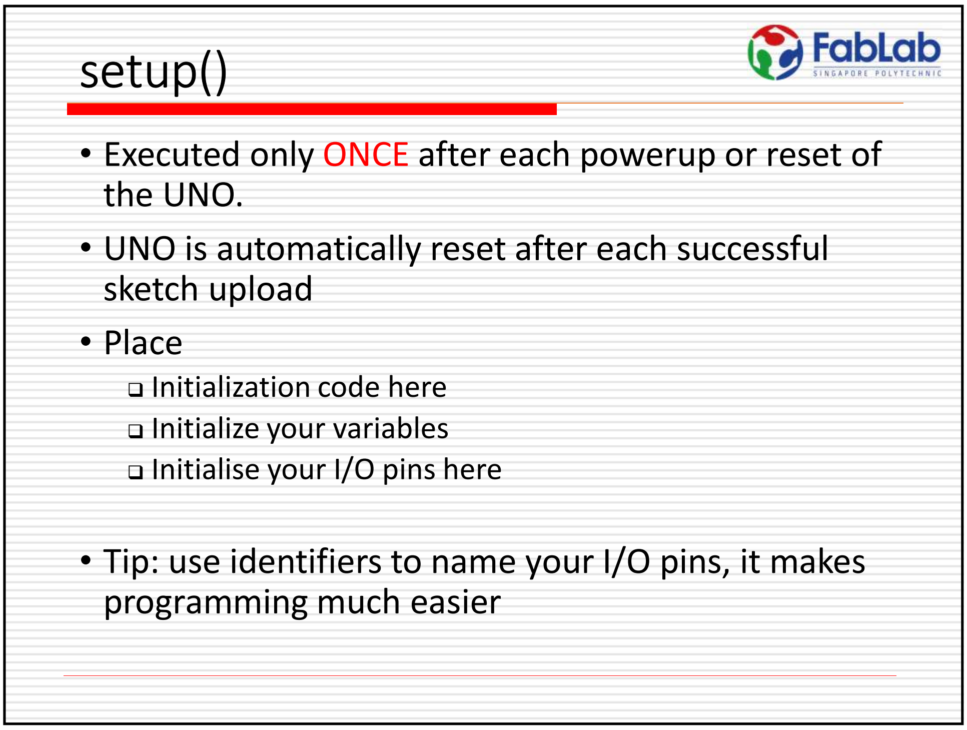
The ARDUINO program



sketch_jun20a

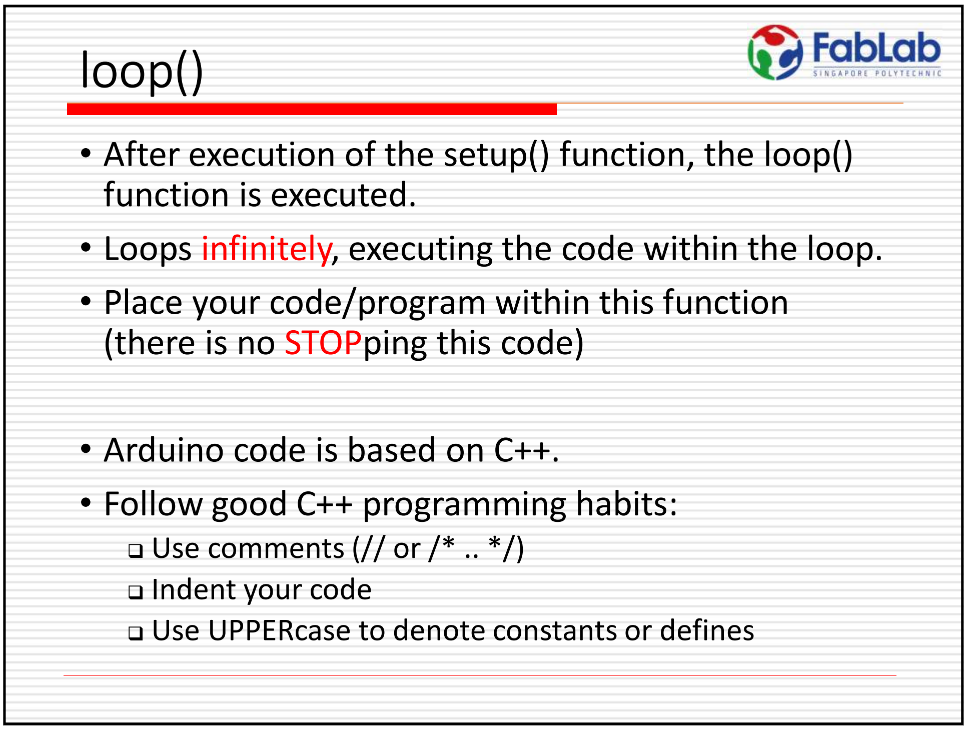
```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

- Called a Sketch (extension .ino)
- Code in the setup function is executed at the start and only once.
- Code in the loop function is executed continually after setup() is run.



setup()

- Executed only ONCE after each powerup or reset of the UNO.
- UNO is automatically reset after each successful sketch upload
- Place
  - ❑ Initialization code here
  - ❑ Initialize your variables
  - ❑ Initialise your I/O pins here
- Tip: use identifiers to name your I/O pins, it makes programming much easier



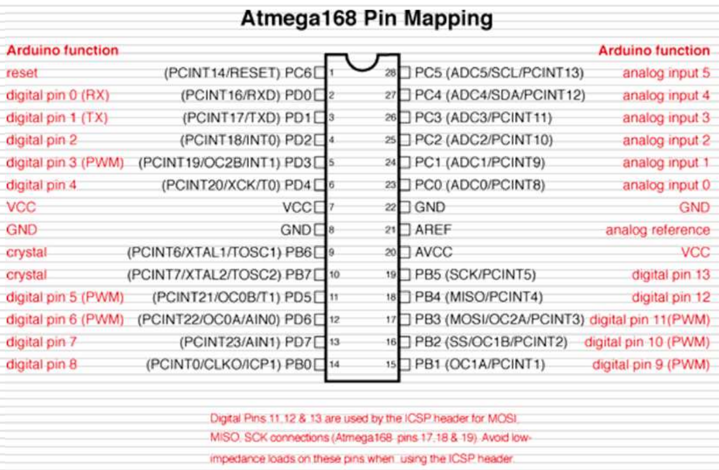
loop()

- After execution of the setup() function, the loop() function is executed.
- Loops infinitely, executing the code within the loop.
- Place your code/program within this function (there is no STOPping this code)
- Arduino code is based on C++.
- Follow good C++ programming habits:
  - ❑ Use comments (// or /* .. */)
  - ❑ Indent your code
  - ❑ Use UPPERcase to denote constants or defines

Digital Input/Output

- ATMega328 has 14 digital input/output ports.
- Digital values (1 = 5V, 0 = 0V)
- Some of these ports are multifunctional, depending on how they are initialised.
- They can perform as
  - ❑ Digital inputs (defaults)
  - ❑ Digital outputs
  - ❑ Pulse-width modulation outputs
- Arduino provides useful library functions for these purposes, simplifying programming.

Arduino pin mapping



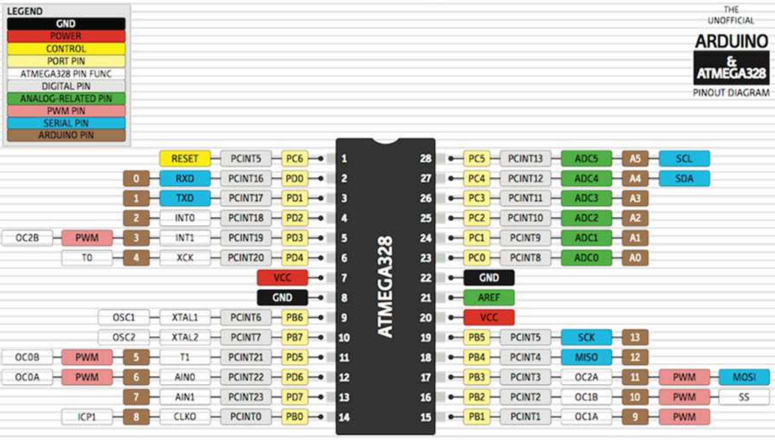
Atmega168 Pin Mapping

Arduino function					Arduino function
reset	(PCINT14/RESET) PC6	1	28	PC5 (ADC5/SCL/PCINT13)	analog input 5
digital pin 0 (RX)	(PCINT16/RXD) PD0	2	27	PC4 (ADC4/SDA/PCINT12)	analog input 4
digital pin 1 (TX)	(PCINT17/TXD) PD1	3	26	PC3 (ADC3/PCINT11)	analog input 3
digital pin 2	(PCINT18/INT0) PD2	4	25	PC2 (ADC2/PCINT10)	analog input 2
digital pin 3 (PWM)	(PCINT19/OC2B/INT1) PD3	5	24	PC1 (ADC1/PCINT9)	analog input 1
digital pin 4	(PCINT20/XCK/T0) PD4	6	23	PC0 (ADC0/PCINT8)	analog input 0
VCC	VCC	7	22	GND	GND
GND	GND	8	21	AREF	analog reference
crystal	(PCINT6/XTAL1/TOSC1) PB6	9	20	AVCC	VCC
crystal	(PCINT7/XTAL2/TOSC2) PB7	10	19	PB5 (SCK/PCINT5)	digital pin 13
digital pin 5 (PWM)	(PCINT21/OC0B/T1) PD5	11	18	PB4 (MISO/PCINT4)	digital pin 12
digital pin 6 (PWM)	(PCINT22/OC0A/AIN0) PD6	12	17	PB3 (MOSI/OC2A/PCINT3)	digital pin 11(PWM)
digital pin 7	(PCINT23/AIN1) PD7	13	16	PB2 (SS/OC1B/PCINT2)	digital pin 10 (PWM)
digital pin 8	(PCINT0/CLKO/ICP1) PB0	14	15	PB1 (OC1A/PCINT1)	digital pin 9 (PWM)

Digital Pins 11,12 & 13 are used by the ICSP header for MOSI, MISO, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.

Ref: Arduino – ATmega328 Pin Mapping

Arduino pin mapping


LEGEND
GND
POWER
CONTROL
PORT PIN
ATMEGA328 PIN FUNC
DIGITAL PIN
ANALOG-RELATED PIN
PWM PIN
SERIAL PIN
ARDUINO PIN
THE UNOFFICIAL
ARDUINO & ATMEGA328
PINOUT DIAGRAM
ATMEGA328
RESET — PCINT5 — PC6 — 1
0 — RXD — PCINT16 — PD0 — 2
1 — TXD — PCINT17 — PD1 — 3
2 — INT0 — PCINT18 — PD2 — 4
OC2B — PWM — 3 — INT1 — PCINT19 — PD3 — 5
T0 — 4 — XCK — PCINT20 — PD4 — 6
VCC — 7
GND — 8
OSC1 — XTAL1 — PCINT6 — PB6 — 9
OSC2 — XTAL2 — PCINT7 — PB7 — 10
OC0B — PWM — 5 — T1 — PCINT21 — PD5 — 11
OC0A — PWM — 6 — AIN0 — PCINT22 — PD6 — 12
7 — AIN1 — PCINT23 — PD7 — 13
ICP1 — 8 — CLKO — PCINT0 — PB0 — 14
28 — PC5 — PCINT13 — ADC5 — A5 — SCL
27 — PC4 — PCINT12 — ADC4 — A4 — SDA
26 — PC3 — PCINT11 — ADC3 — A3
25 — PC2 — PCINT10 — ADC2 — A2
24 — PC1 — PCINT9 — ADC1 — A1
23 — PC0 — PCINT8 — ADC0 — A0
22 — GND
21 — AREF
20 — VCC
19 — PB5 — PCINT5 — SCK — 13
18 — PB4 — PCINT4 — MISO — 12
17 — PB3 — PCINT3 — OC2A — 11 — PWM — MOSI
16 — PB2 — PCINT2 — OC1B — 10 — PWM — SS
15 — PB1 — PCINT1 — OC1A — 9 — PWM

www.pighixxx.com

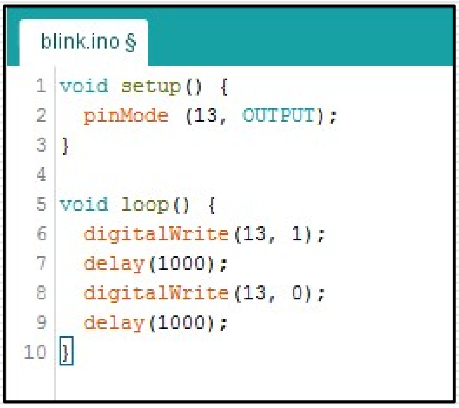
29 JAN 2013

Tip: Use the BROWN identifiers for your Arduino sketch

Ref: Pighxxx ATMega328 Pinout

Digital Output

- pinmode()
  Initialise digital pin 13 to be a output port
- Repeat
  - digitalWrite()
    Turn ON the LED
  - delay()
    Wait 1 second
  - Turn OFF the LED
  - Wait 1 second
- Arduino Programming reference

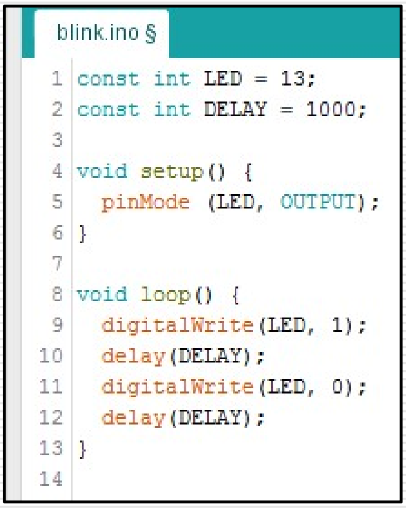


```
void setup() {
  pinMode (13, OUTPUT);
}

void loop() {
  digitalWrite(13, 1);
  delay(1000);
  digitalWrite(13, 0);
  delay(1000);
}
```

Colour coding helps in recognizing in-built functions, reserved words, values

Using identifiers

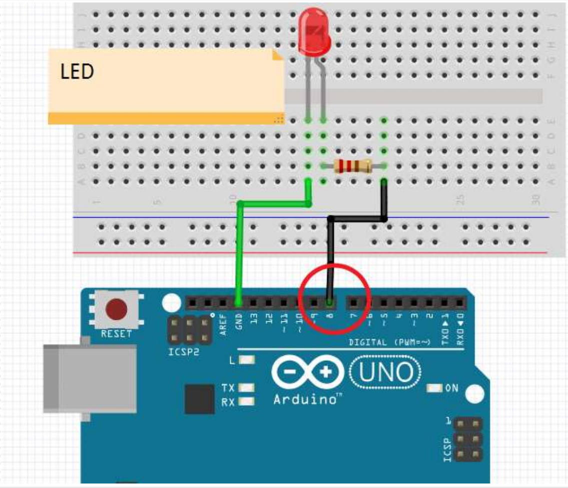
- Name the ports that you use, it makes it easier to change, configure, understand.
- Examine the following code.
  - ❑ How do I change the port from 13 to 4?
  - ❑ How do I change the delay to 0.5sec ?



```
const int LED = 13;
const int DELAY = 1000;

void setup() {
  pinMode (LED, OUTPUT);
}

void loop() {
  digitalWrite(LED, 1);
  delay(DELAY);
  digitalWrite(LED, 0);
  delay(DELAY);
}

```

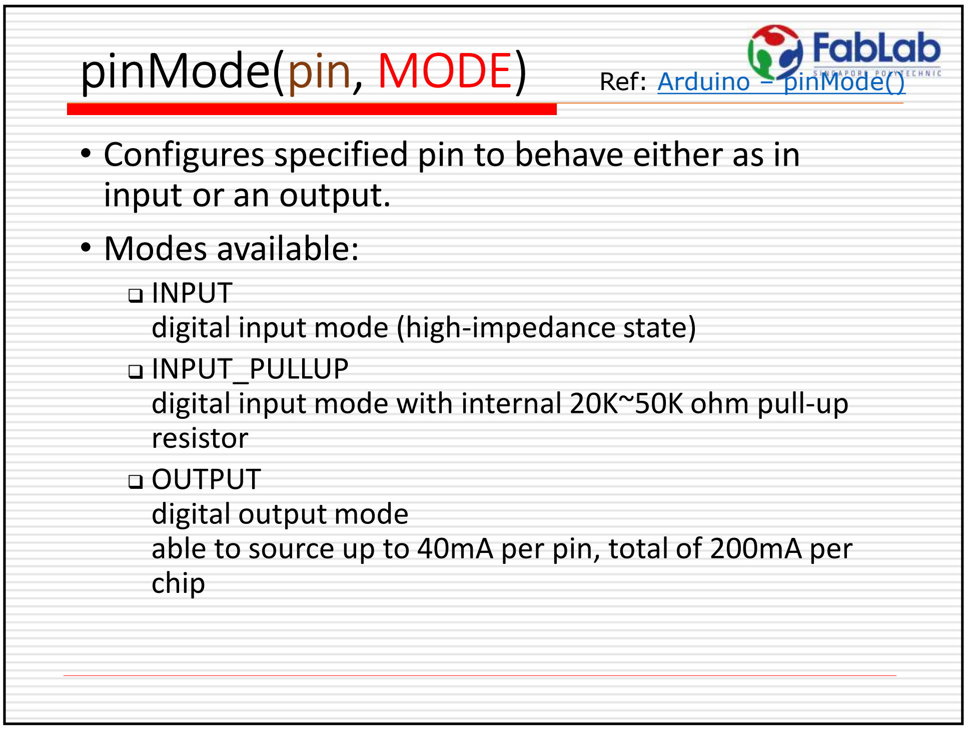
UNO Port 13 is wired to a LED, useful for testing!

Adding an external LED



1x	LED
1x	220 ohm R
2x	Wires

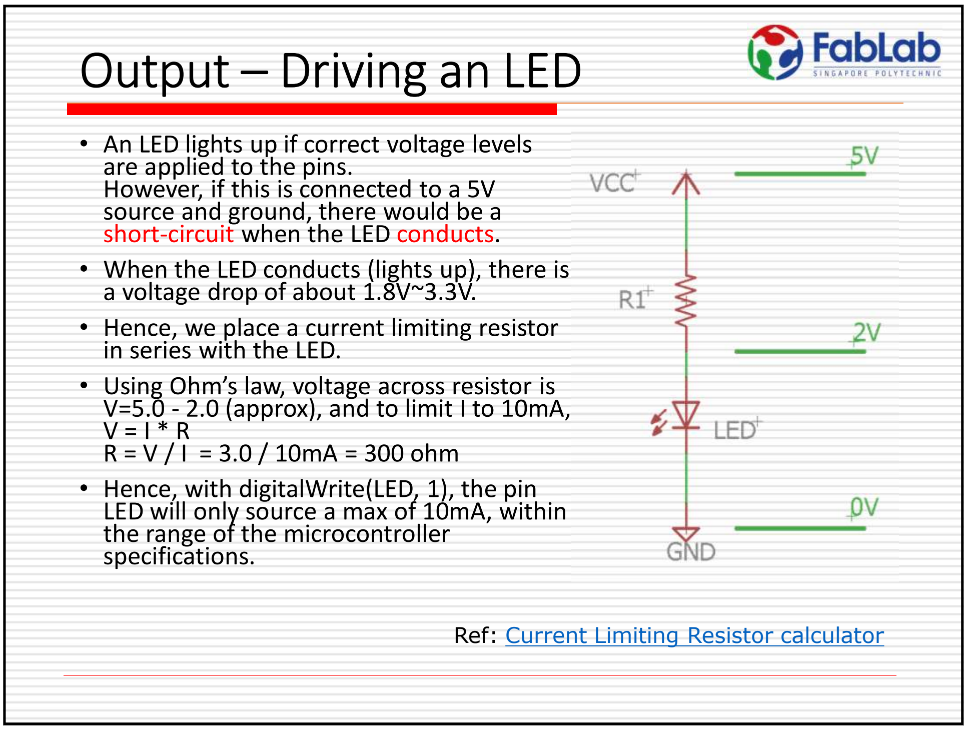
LED anode is connected to PD8



pinMode(pin, MODE)

Ref: Arduino – pinMode()

- Configures specified pin to behave either as in input or an output.
- Modes available:
  - ❑ INPUT
    digital input mode (high-impedance state)
  - ❑ INPUT_PULLUP
    digital input mode with internal 20K~50K ohm pull-up resistor
  - ❑ OUTPUT
    digital output mode
    able to source up to 40mA per pin, total of 200mA per chip



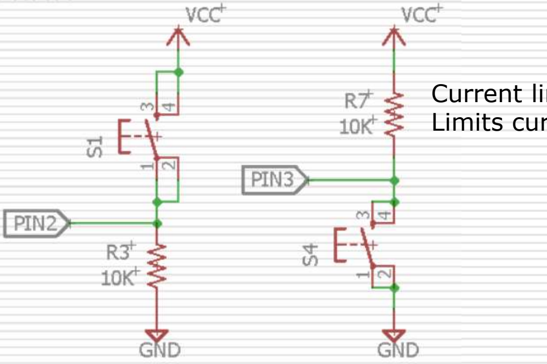
Output – Driving an LED

- An LED lights up if correct voltage levels are applied to the pins. However, if this is connected to a 5V source and ground, there would be a short-circuit when the LED conducts.
- When the LED conducts (lights up), there is a voltage drop of about 1.8V~3.3V.
- Hence, we place a current limiting resistor in series with the LED.
- Using Ohm's law, voltage across resistor is V=5.0 - 2.0 (approx), and to limit I to 10mA,
  V = I * R
  R = V / I = 3.0 / 10mA = 300 ohm
- Hence, with digitalWrite(LED, 1), the pin LED will only source a max of 10mA, within the range of the microcontroller specifications.

Ref: Current Limiting Resistor calculator

Digital input

- We can read digital inputs using the pins.
- This is done by connecting a switch to a limiting resistor and monitoring the status.

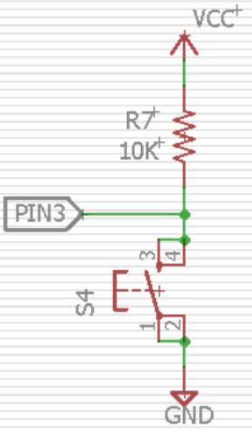


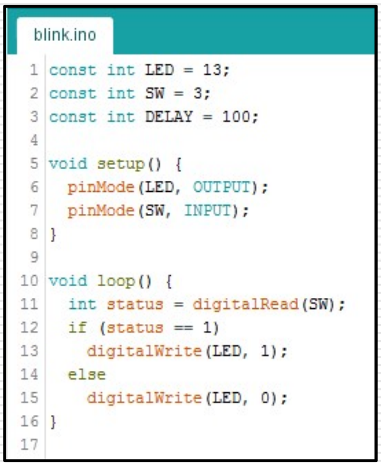
Current limiting resistor 10K
Limits current to 0.5mA

Switch OFF, PIN2=0
Switch ON, PIN2=1

Switch OFF, PIN3=1
Switch ON, PIN3=0

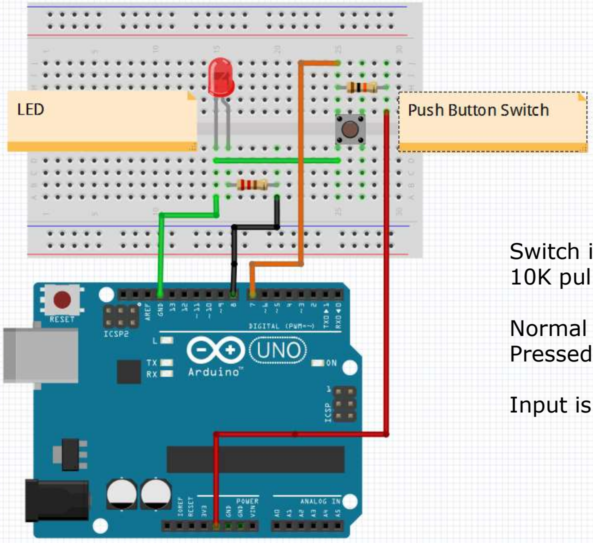
Read Switch display status on LED





```
const int LED = 13;
const int SW = 3;
const int DELAY = 100;

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(SW, INPUT);
}

void loop() {
  int status = digitalRead(SW);
  if (status == 1)
    digitalWrite(LED, 1);
  else
    digitalWrite(LED, 0);
}
```

Adding an input switch



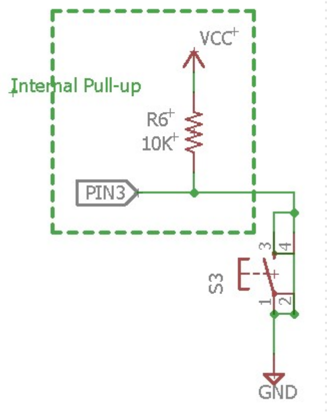
Switch is added with
10K pull-up resistor.

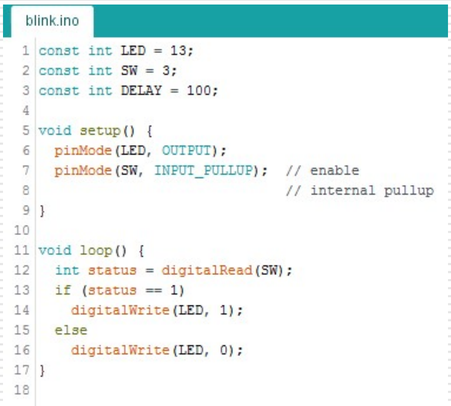
Normal reading = HIGH
Pressed = LOW

Input is read using PD7

Using internal pull-up

- We can also use the internal pull-up resistors. Only need external switch.



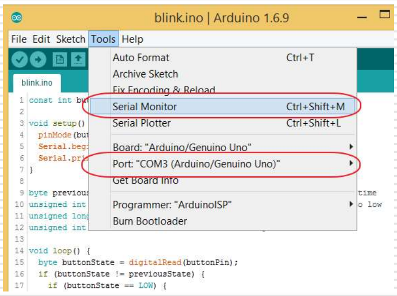


```
const int LED = 13;
const int SW = 3;
const int DELAY = 100;

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(SW, INPUT_PULLUP);  // enable
                              // internal pullup
}

void loop() {
  int status = digitalRead(SW);
  if (status == 1)
    digitalWrite(LED, 1);
  else
    digitalWrite(LED, 0);
}

```

Ref: YouTube – Arduino Internal Pull-up

Serial mode debugging

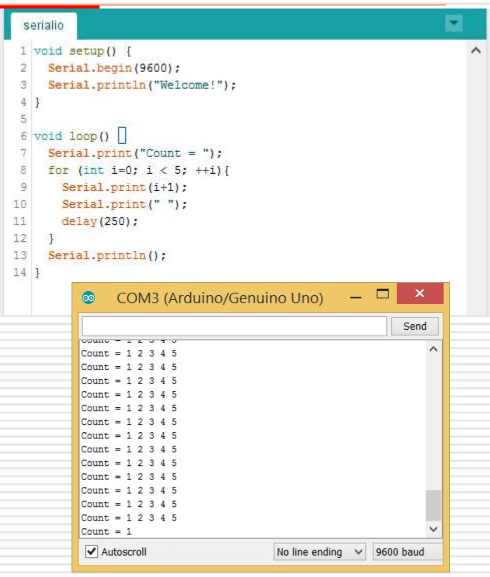
- Arduino systems have a unique method of troubleshooting by using the Serial Interface.
- UNO uses pins 0 and 1 for serial receive and transmit via the USB interface.
- We need to
  - ❑ Set the Serial Port to use
  - ❑ Turn on the Serial Monitor
  - ❑ Set the baud (TX/RX rate)


blink.ino | Arduino 1.6.9
File Edit Sketch Tools Help
blink.ino
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
Board: "Arduino/Genuino Uno"
Port: "COM3 (Arduino/Genuino Uno)"
Get Board Info
Programmer: "ArduinoISP"
Burn Bootloader

```
const int bu

void setup()
  pinMode(bu
  Serial.beg
  Serial.pri
}

byte previou                                        time
unsigned int                                        o low
unsigned lon
unsigned int

void loop() {
  byte buttonState = digitalRead(buttonPin);
  if (buttonState != previousState) {
    if (buttonState == LOW) {
```

Serial mode test

- Initialise the serial port
  - ❑ Serial.begin(<baud rate>)
  - ❑ Typical: 9600 or 57600 this is the speed of the link
- Send data to serial port
  - ❑ Serial.print()
  - ❑ Serial.println()
  - ❑ If you see garbage then your speed is set incorrectly
- Example shows how a message and numbers 1 to 5 are output to the serial monitor.



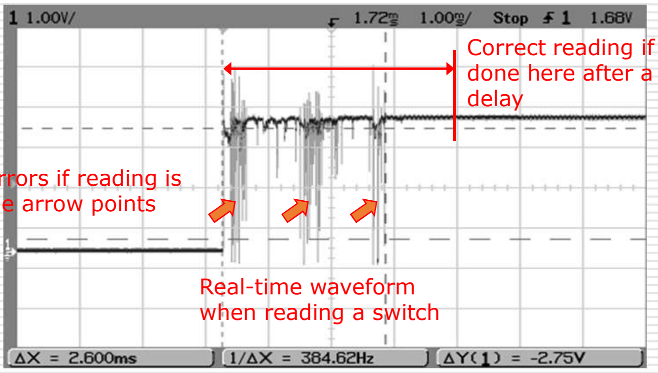
```
void setup() {
  Serial.begin(9600);
  Serial.println("Welcome!");
}

void loop() {
  Serial.print("Count = ");
  for (int i=0; i < 5; ++i){
    Serial.print(i+1);
    Serial.print(" ");
    delay(250);
  }
  Serial.println();
}
```

```
Count = 1 2 3 4 5
Count = 1 2 3 4 5
Count = 1 2 3 4 5
Count = 1 2 3 4 5
Count = 1 2 3 4 5
Count = 1 2 3 4 5
Count = 1 2 3 4 5
Count = 1 2 3 4 5
Count = 1 2 3 4 5
Count = 1 2 3 4 5
Count = 1 2 3 4 5
Count = 1 2 3 4 5
Count = 1 2 3 4 5
Count = 1
```

☑ Autoscroll   No line ending   9600 baud

Problems with reading mechanical switches

- A switch is a mechanical device, when pressed, it creates transients (bouncing), which causes incorrect states to be read.



Correct reading if done here after a delay

Possible errors if reading is done at the arrow points

Real-time waveform when reading a switch

Ref: Wikipedia - Switch

Can use software to "debounce" i.e. remove the bouncing effects of the switch

Debounce code

- We can correct the error by reading the switch (again) after a short delay (e.g. 25 ms)
  This correction is called debouncing.

Embed:
Video: Android Button Tutorial How to debounce a button using an Arduino Uno or Mega

Video: YouTube Arduino Button Tutorial

Counting switch presses

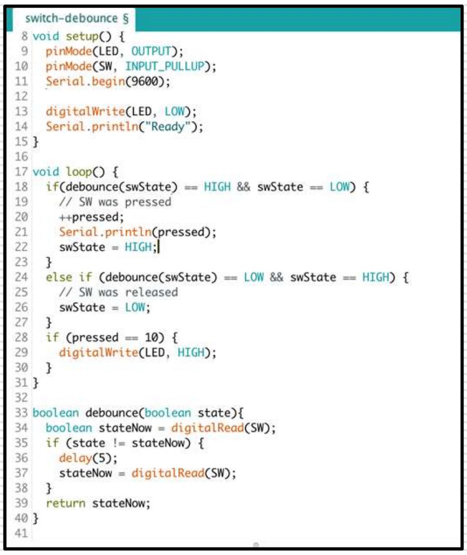
- Counts the number of times a switch is pressed, after 10 counts, the LED is lit up.
  Serial interface helps in showing us that the code is working correctly
- Algorithm

```
Initialise pins, serial, led
Loop
  count = 0
  while count <> 10 do
    if switch goes from LOW to HIGH then
      increment count
      display count using serial
      switch off LED
    endif
  end-while
  light up LED
end-loop
```

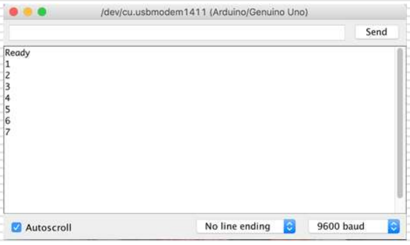
Convert the code into an Arduino sketch and test it!

Ref: Arduino - debounce

debounce

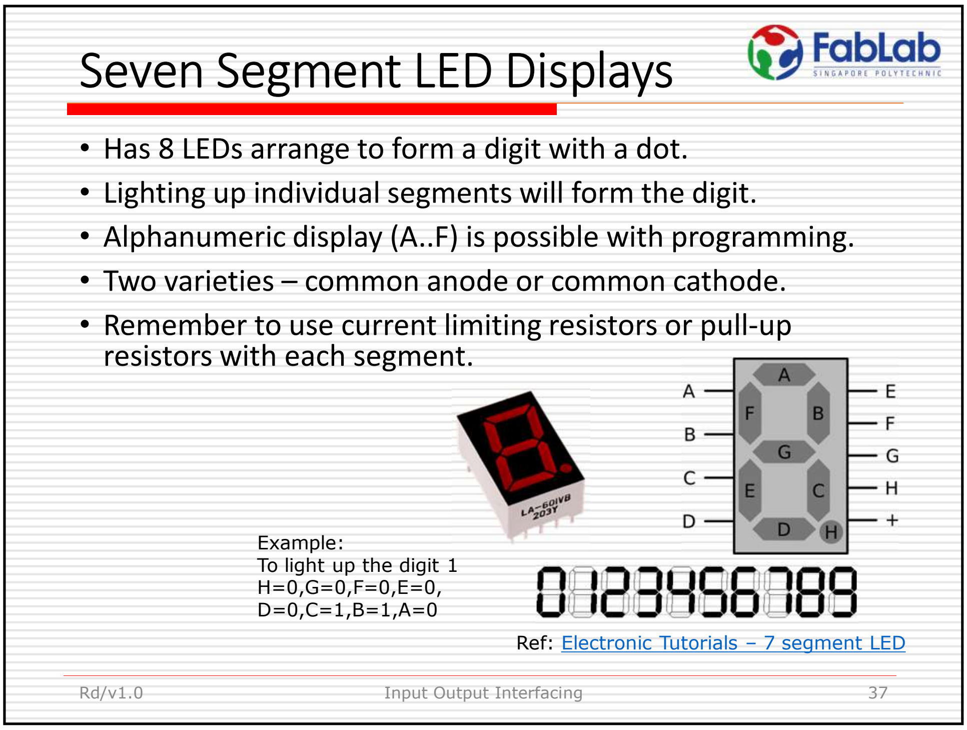


```
void setup() {
  pinMode(LED, OUTPUT);
  pinMode(SW, INPUT_PULLUP);
  Serial.begin(9600);

  digitalWrite(LED, LOW);
  Serial.println("Ready");
}

void loop() {
  if(debounce(swState) == HIGH && swState == LOW) {
    // SW was pressed
    ++pressed;
    Serial.println(pressed);
    swState = HIGH;
  }
  else if (debounce(swState) == LOW && swState == HIGH) {
    // SW was released
    swState = LOW;
  }
  if (pressed == 10) {
    digitalWrite(LED, HIGH);
  }
}

boolean debounce(boolean state){
  boolean stateNow = digitalRead(SW);
  if (state != stateNow) {
    delay(5);
    stateNow = digitalRead(SW);
  }
  return stateNow;
}

```



```
Ready
1
2
3
4
5
6
7
```

Debounce code prevents multiple reading of the switch making each switch press react positively



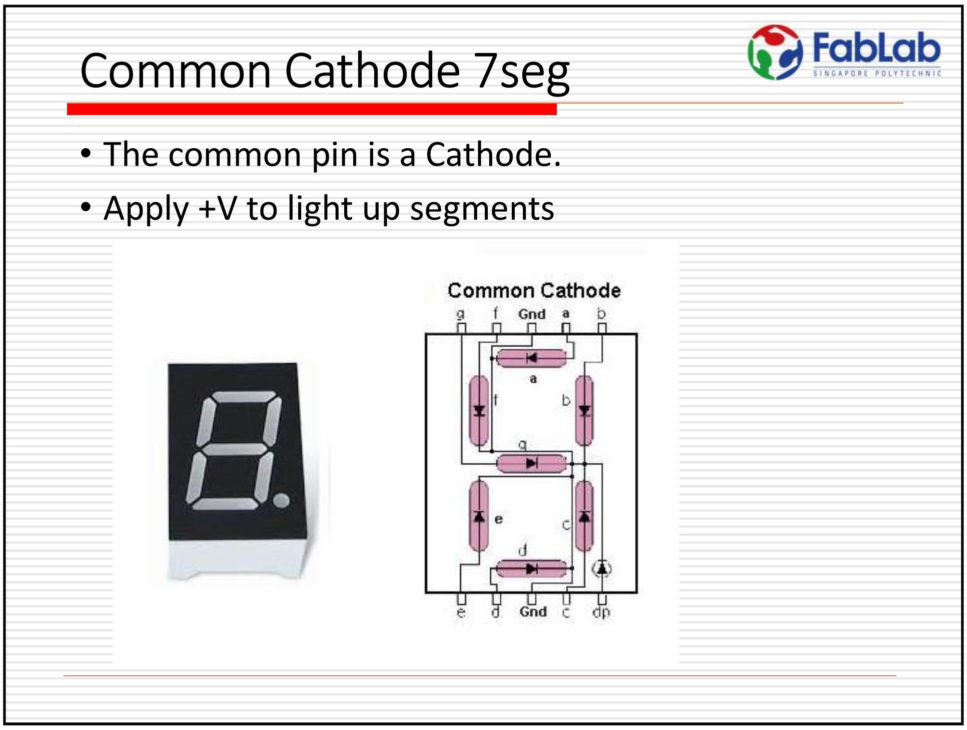
Seven Segment LED Displays

- Has 8 LEDs arrange to form a digit with a dot.
- Lighting up individual segments will form the digit.
- Alphanumeric display (A..F) is possible with programming.
- Two varieties – common anode or common cathode.
- Remember to use current limiting resistors or pull-up resistors with each segment.

Example:
To light up the digit 1
H=0,G=0,F=0,E=0,
D=0,C=1,B=1,A=0

LA-601VB
203Y
A
B
C
D
E
F
G
H
+
A
F
B
G
E
C
D
H
0123456789

Ref: Electronic Tutorials – 7 segment LED

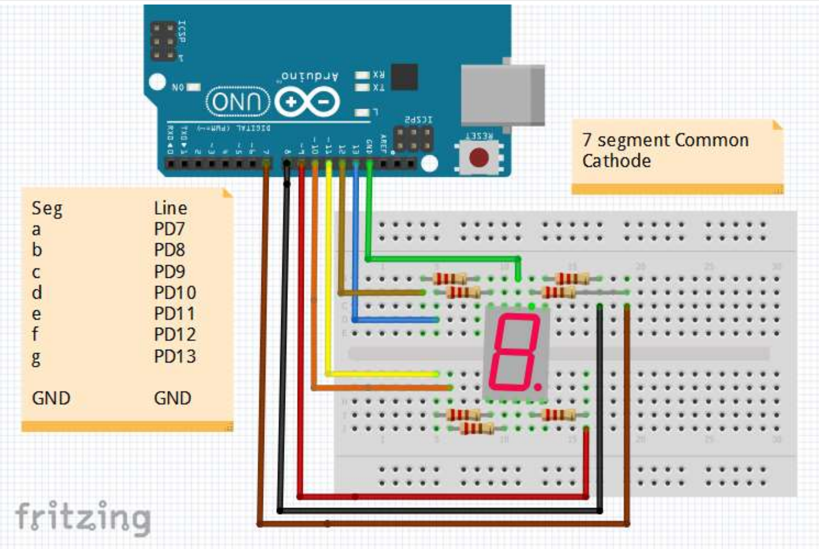


Common Cathode 7seg

- The common pin is a Cathode.
- Apply +V to light up segments

Common Cathode
g
f
Gnd
a
b
a
f
b
g
e
c
d
e
d
Gnd
c
dp

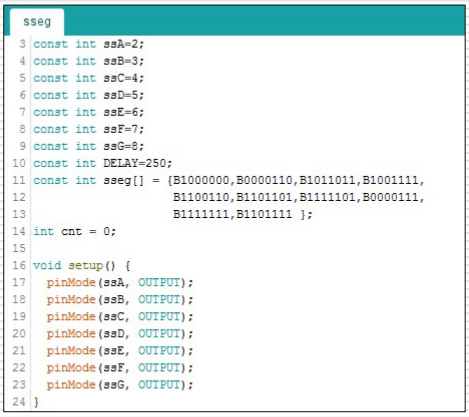
Wiring up a 7 segment



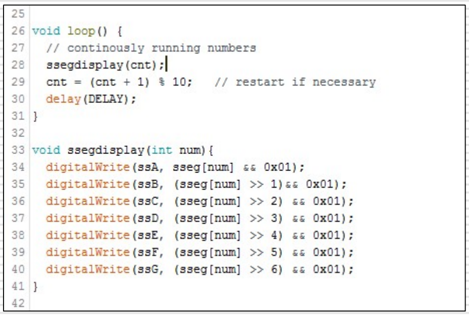
7 segment Common Cathode

Seg	Line
a	PD7
b	PD8
c	PD9
d	PD10
e	PD11
f	PD12
g	PD13
GND	GND

Seven Segment code



```
const int ssA=2;
const int ssB=3;
const int ssC=4;
const int ssD=5;
const int ssE=6;
const int ssF=7;
const int ssG=8;
const int DELAY=250;
const int sseg[] = {B1000000,B0000110,B1011011,B1001111,
                    B1100110,B1101101,B1111101,B0000111,
                    B1111111,B1101111 };
int cnt = 0;

void setup() {
  pinMode(ssA, OUTPUT);
  pinMode(ssB, OUTPUT);
  pinMode(ssC, OUTPUT);
  pinMode(ssD, OUTPUT);
  pinMode(ssE, OUTPUT);
  pinMode(ssF, OUTPUT);
  pinMode(ssG, OUTPUT);
}
```



```

void loop() {
  // continously running numbers
  ssegdisplay(cnt);
  cnt = (cnt + 1) % 10;   // restart if necessary
  delay(DELAY);
}

void ssegdisplay(int num){
  digitalWrite(ssA, sseg[num] && 0x01);
  digitalWrite(ssB, (sseg[num] >> 1)&& 0x01);
  digitalWrite(ssC, (sseg[num] >> 2) && 0x01);
  digitalWrite(ssD, (sseg[num] >> 3) && 0x01);
  digitalWrite(ssE, (sseg[num] >> 4) && 0x01);
  digitalWrite(ssF, (sseg[num] >> 5) && 0x01);
  digitalWrite(ssG, (sseg[num] >> 6) && 0x01);
}

```

This code uses 7 digitial I/O lines.
Can you think of ways of reducing
the number of I/O lines used?

Getting Started with Arduino

Rodney Dorville