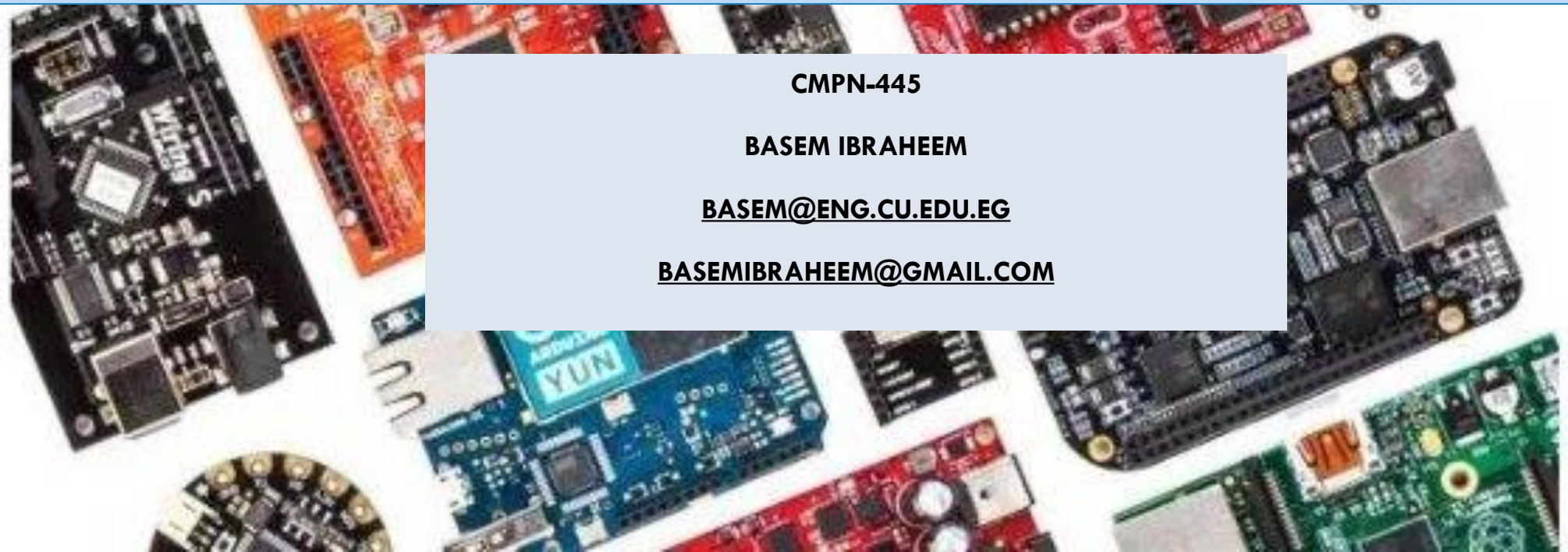
SELECTING YOUR MICROCONTROLLER



CMPN-445

BASEM IBRAHEEM

BASEM@ENG.CU.EDU.EG

BASEMIBRAHEEM@GMAIL.COM

AGENDA

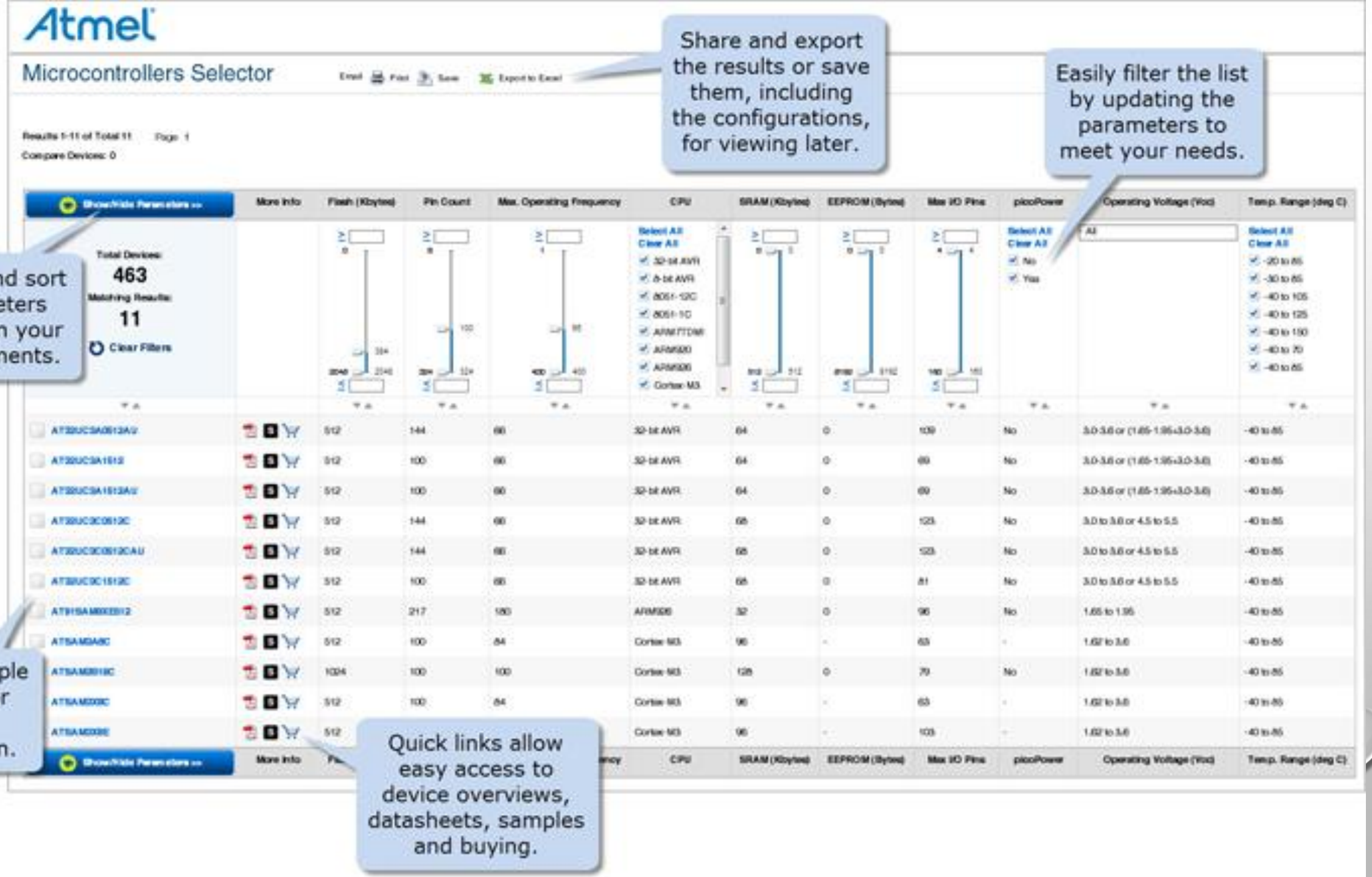
- BASES FOR SELECTING A MICROCONTROLLER
- CORTEX A-R-M
- SELECTOR TOOLS
- INTRODUCTION TO ARDUINO
  - WHY ARDUINO
  - SHIELDS
  - CONNECTION CONSIDERATIONS
  - CONSTRUCTION

BASES FOR SELECTING A MICROCONTROLLER

- DO I NEED A MICROCONTROLLER OR A MICROPROCESSOR ?
  - ONBOARD WI-FI / BLUETOOTH / FLASH / LARGE I/O / GRAPHICS UNIT
  - SPI / I2C / UART
- BITS SIZE (8/16/32) : DATA BUS , REGISTER SIZE , PORT SIZE – ADDRESS BUS (PAGING REGISTER)
- OSCILLATOR FREQUENCY
- PACKAGING FOR SIZE AND ACCESSIBILITY

BASES FOR SELECTING A MICROCONTROLLER

- POWER CONSUMPTION
- RAM / ROM SIZES
- I/O PORTS
- COST PER UNIT
- MANUFACTURE AVAILABILITY
- DEVELOPMENT PLATFORM / INSTRUCTION SET / SIMULATOR / EMULATOR
- OPERATING ENVIRONMENT : MOBILE / FIXED , STABLE / SHOCKS , TEMPERATURE , MOIST

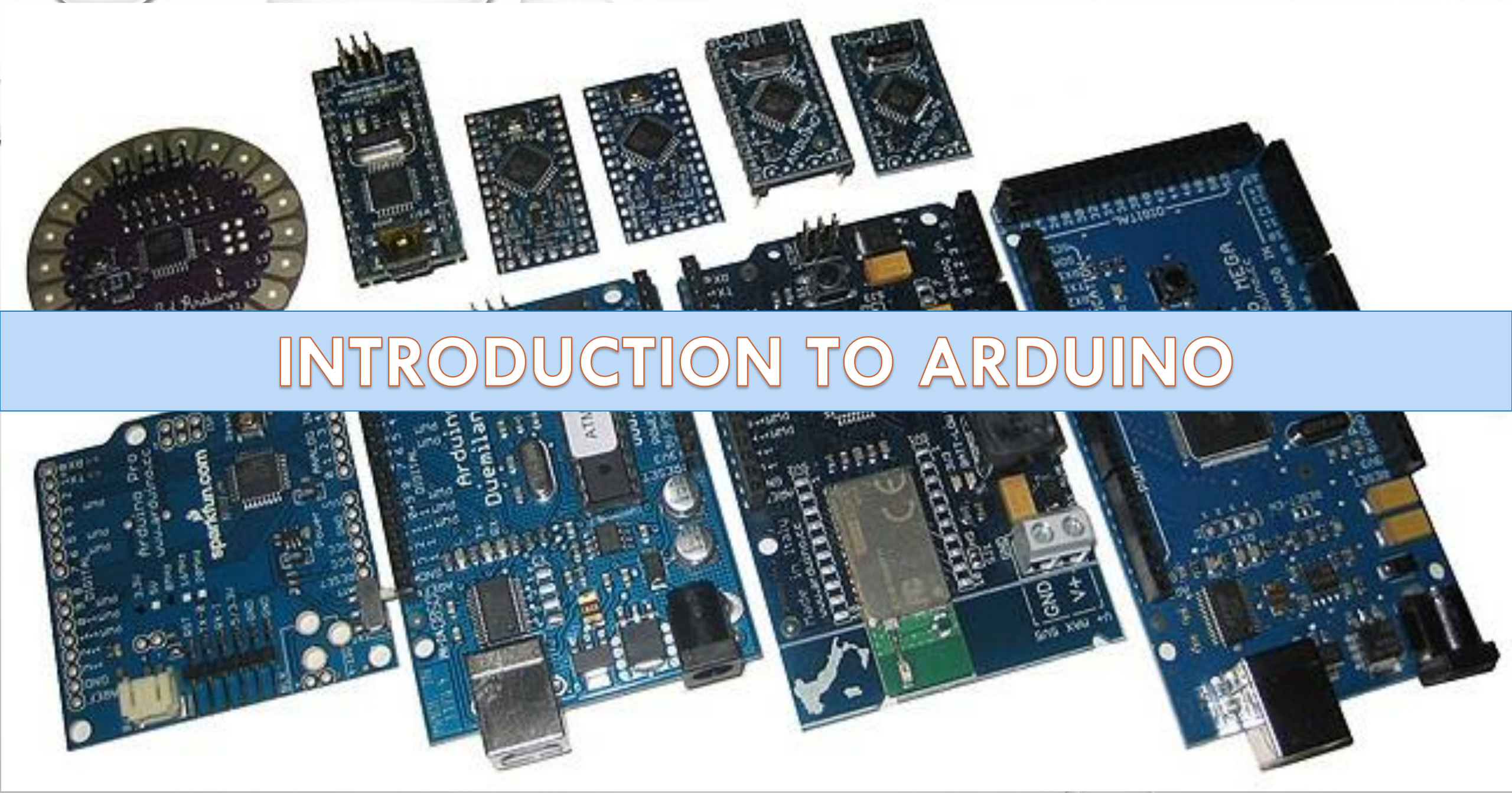

Atmel
Microcontrollers Selector
Email
Print
Save
Export to Excel
Share and export the results or save them, including the configurations, for viewing later.
Easily filter the list by updating the parameters to meet your needs.
Select and sort parameters based on your requirements.
Select multiple devices for easy comparison.
Quick links allow easy access to device overviews, datasheets, samples and buying.
Results 1-11 of Total 11
Page 1
Compare Devices: 0
Show/Hide Parameters
More Info
Flash (Kbytes)
Pin Count
Max. Operating Frequency
CPU
SRAM (Kbytes)
EEPROM (Bytes)
Max I/O Pins
picoPower
Operating Voltage (Vcc)
Temp. Range (deg C)
Total Devices:
463
Matching Results:
11
Clear Filters
Select All
Clear All
32-bit AVR
8-bit AVR
8051-12C
8051-1C
ARM7TDMI
ARM920
ARM926
Cortex-M3
No
Yes
All
-20 to 85
-30 to 85
-40 to 105
-40 to 125
-40 to 150
-40 to 70
-40 to 85
AT32UC3A0512A/U
AT32UC3A1512
AT32UC3A1512A/U
AT32UC3C0512C
AT32UC3C0512CAU
AT32UC3C1512C
AT91SAM9XE512
AT5AMGA5C
AT5AMG51C
AT5AM33C
AT5AM33E

CORTEX A – R – M BY SILICON LABS

- CORTEX A
  - MICROPROCESSOR FOR PERFORMANCE INTENSIVE SYSTEMS
  - RUN AND OS SUCH AS ANDROID
  - MULTIPLE THREADED
  - COMPLEX HIGHSPEED INTERFACES : ETHERNET - MICROSD - USB - ON BOARD WI-FI
  - RASPBERRY PI 2 / SMARTPHONES

CORTEX A – R – M BY SILICON LABS

- CORTEX R
  - MICROPROCESSOR FOR HIGH PERFORMANCE REAL-TIME APPLICATIONS
  - BLUE-RAY – AIRBAGS – BRAKING SYSTEM – ENGINE MANAGEMENT
- CORTEX M
  - MICROCONTROLLER FOR EMBEDDED APPLICATIONS
  - RUN YOUR CODE OR BARE CODE AS BOOT LOADER
  - (0) SINGLE THREADED
  - ONBOARD I2C , SPI , UART , USB
  - YOUR CODE SHOULD HANDLE ALL ACTIVITIES IN A SINGLE THREAD (CODING IS HARDER)



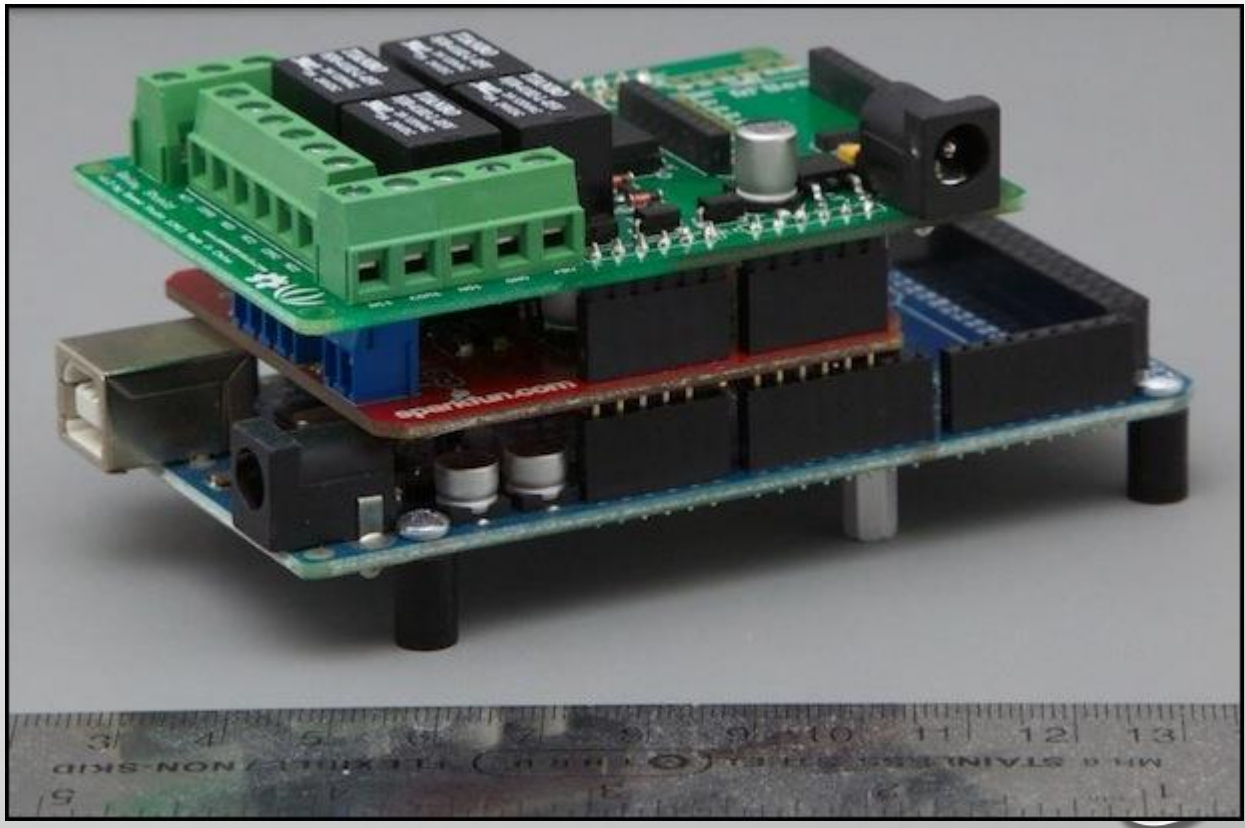
INTRODUCTION TO ARDUINO

WHY ARDUINO ?

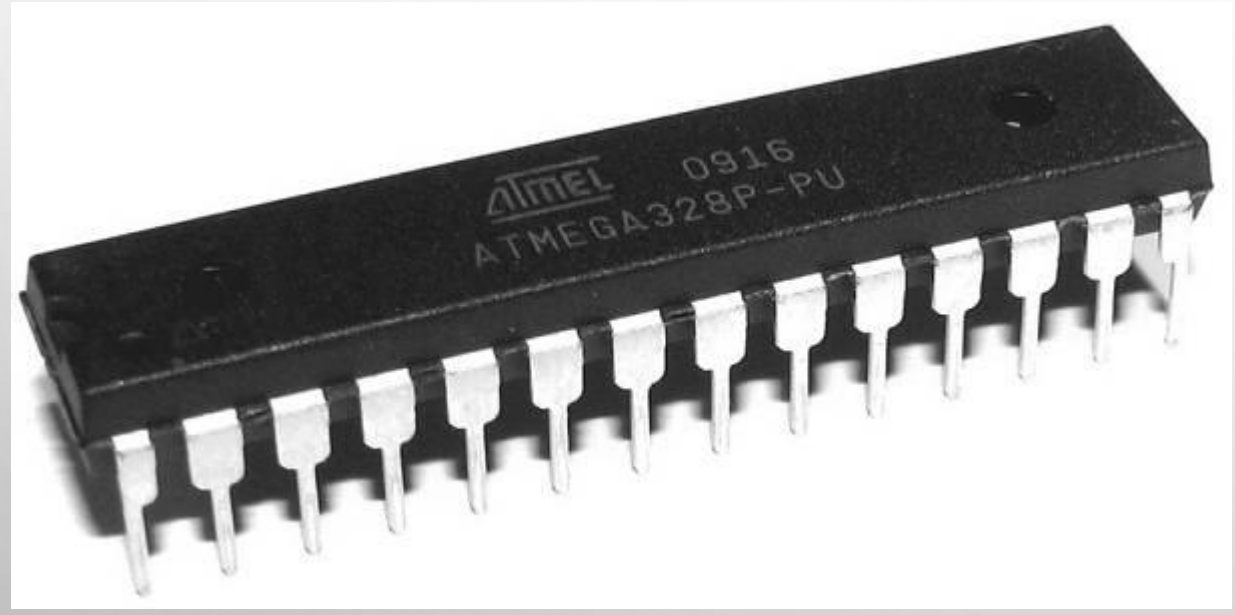
- IMPROVEMENT OVER OTHER MICRO-CONTROLLERS FOR BEGINNER USAGE
- INEXPENSIVE
- IDE ON MULTIPLE PLATFORMS (MAC, PC, LINUX)
- SIMPLE, CLEAR PROGRAMMING ENVIRONMENT: “C”
- OPEN SOURCE AND EXTENSIBLE SOFTWARE/HARDWARE
- ARDUINO SHIELDS
- COMPATIBLE WITH EMBEDDED SYSTEM’S COURSE LABS

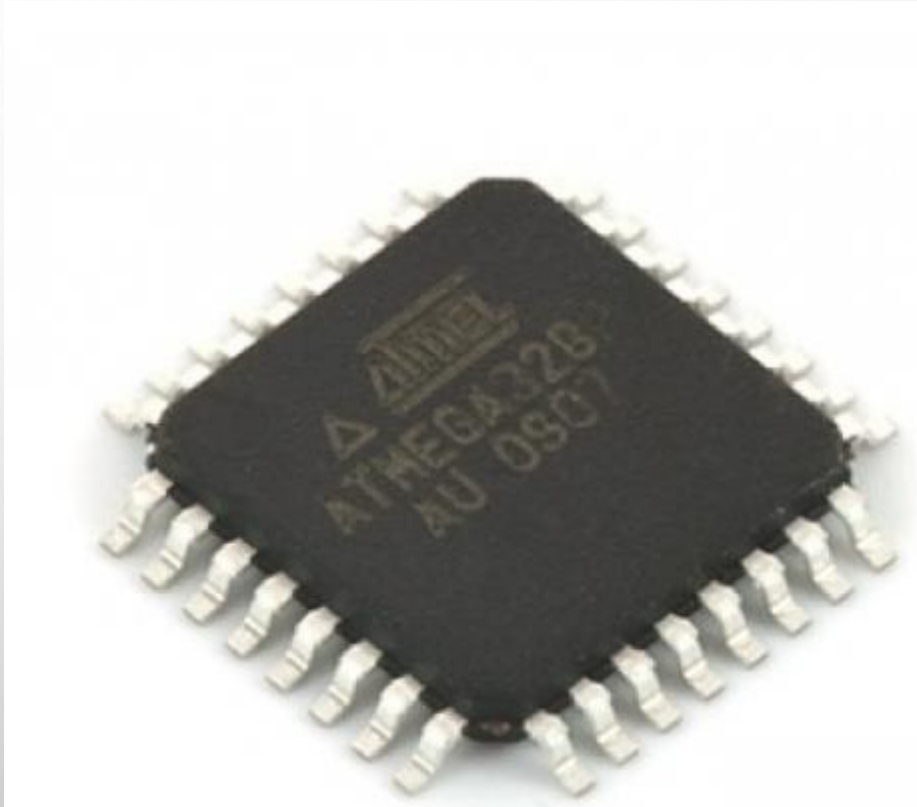
ARDUINO SHIELDS

- VGA CAMERA
- GPS
- LCD DISPLAYS
- ETHERNET
- WIFI
- MOTOR CONTROL (DC, STEPPER)
- Load Cell
- Accelerometer/ Gyros
- LED displays
- Memory Cards
- Weather Sensors
- Relays

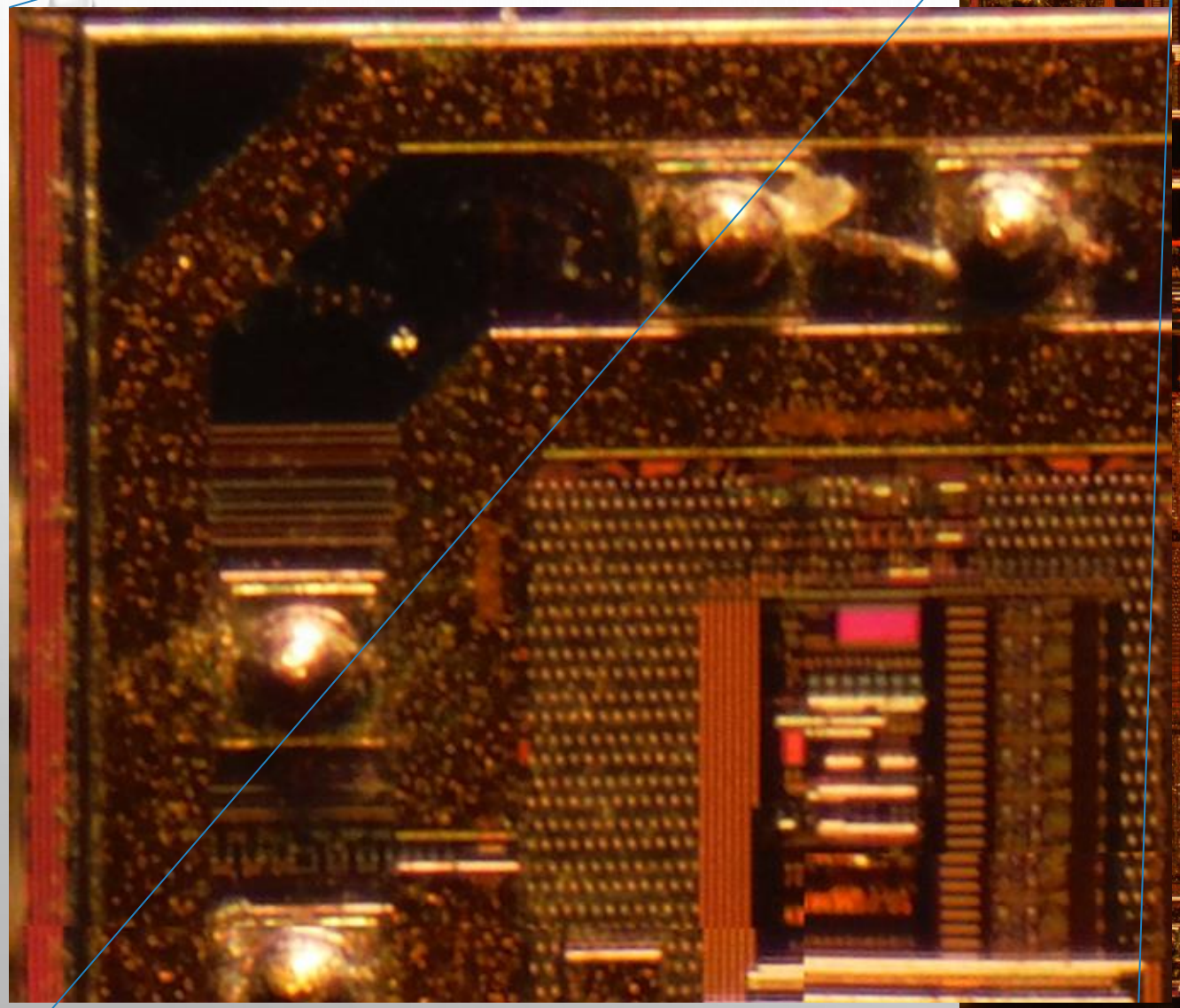


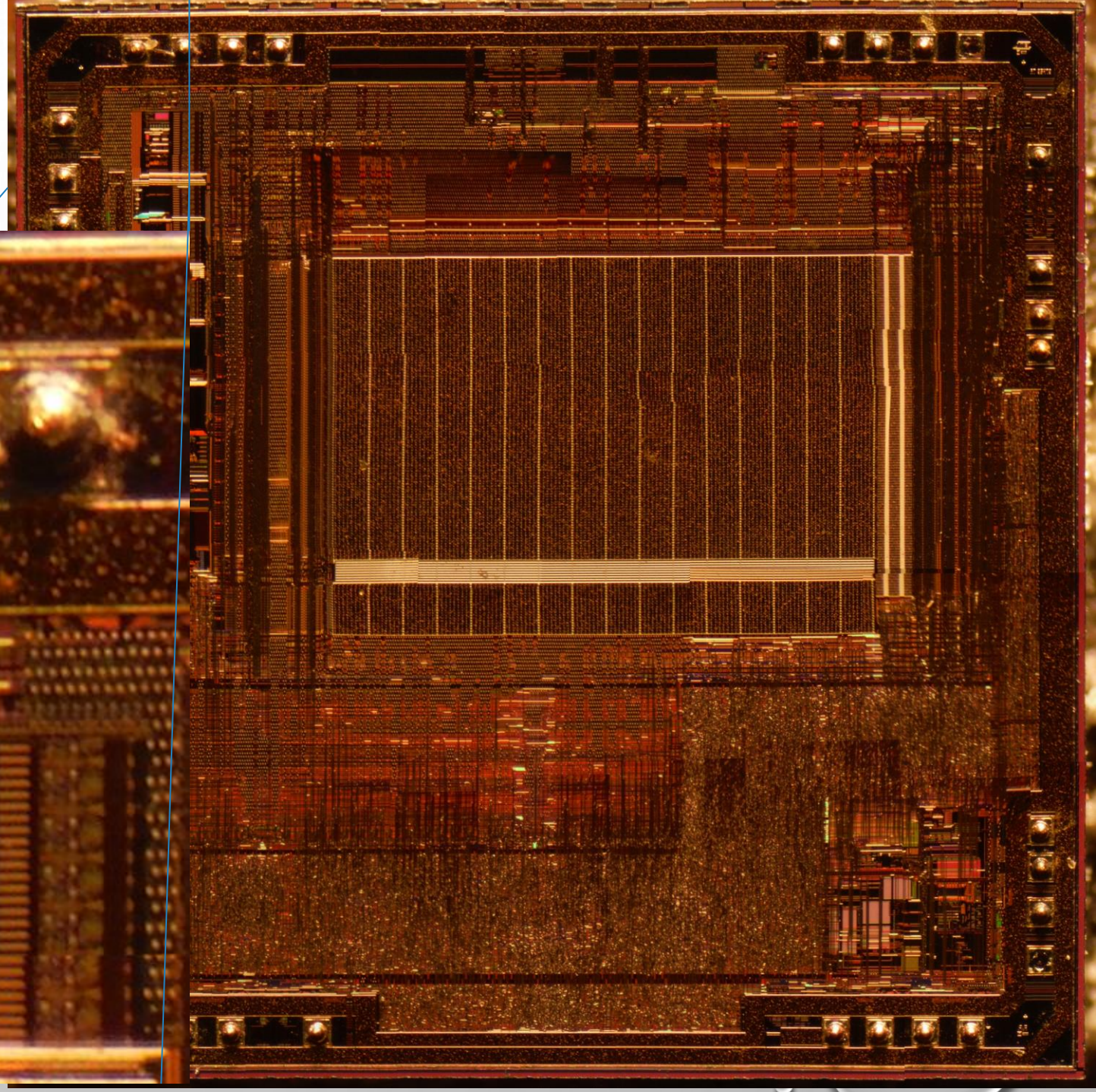
CONSTRUCTION : ATMEGA328P DIP / QFP



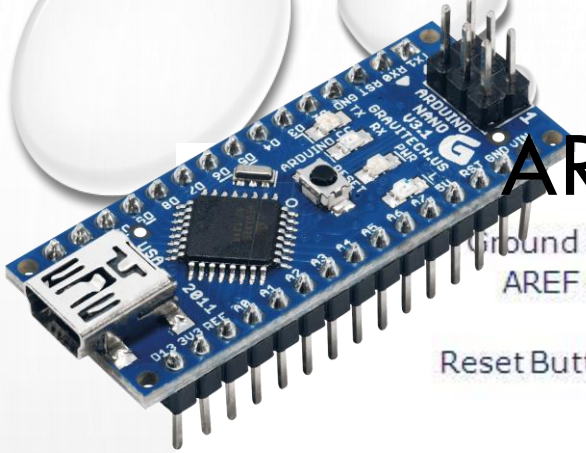


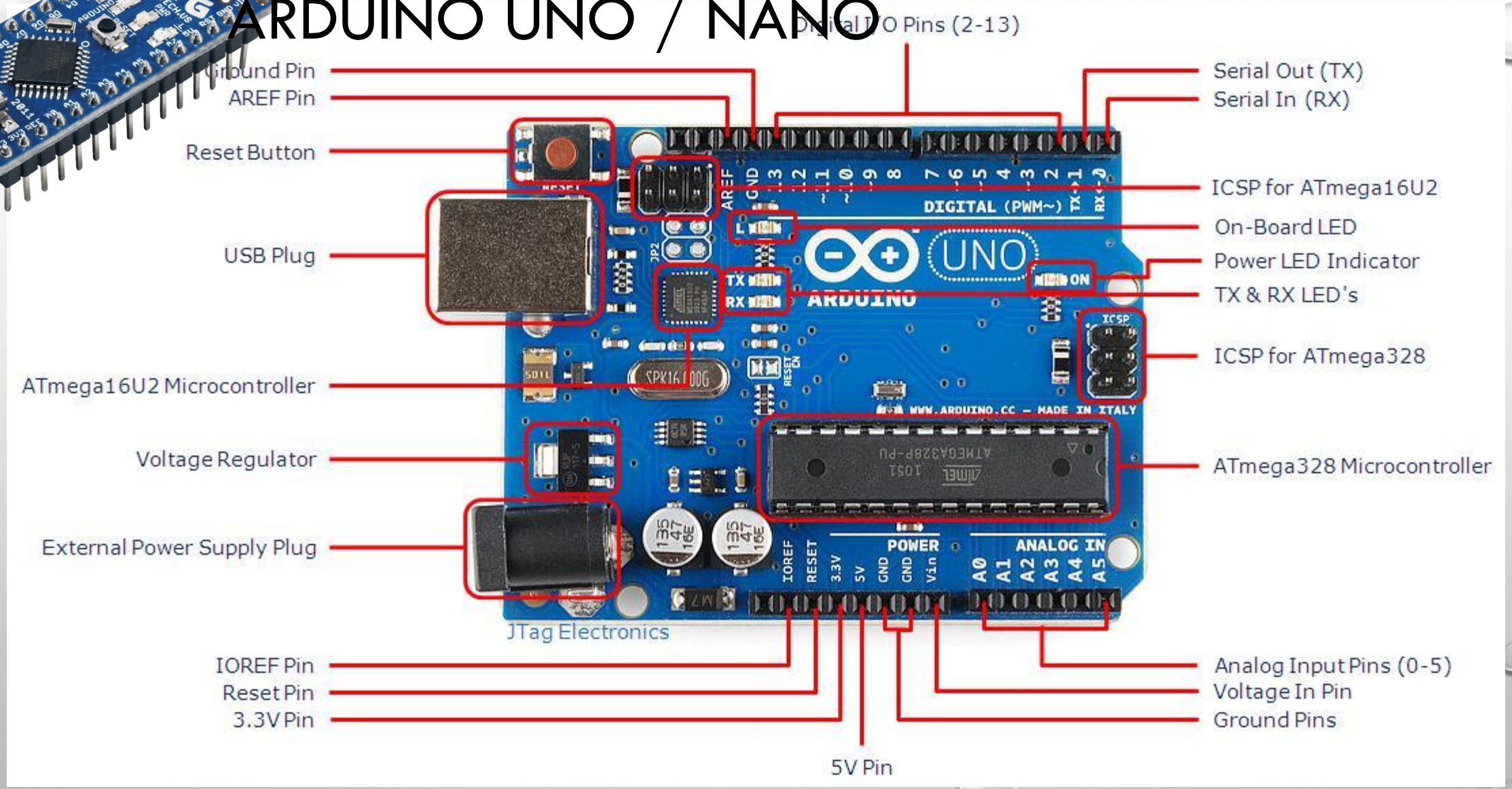
ATMEGA328P QFP

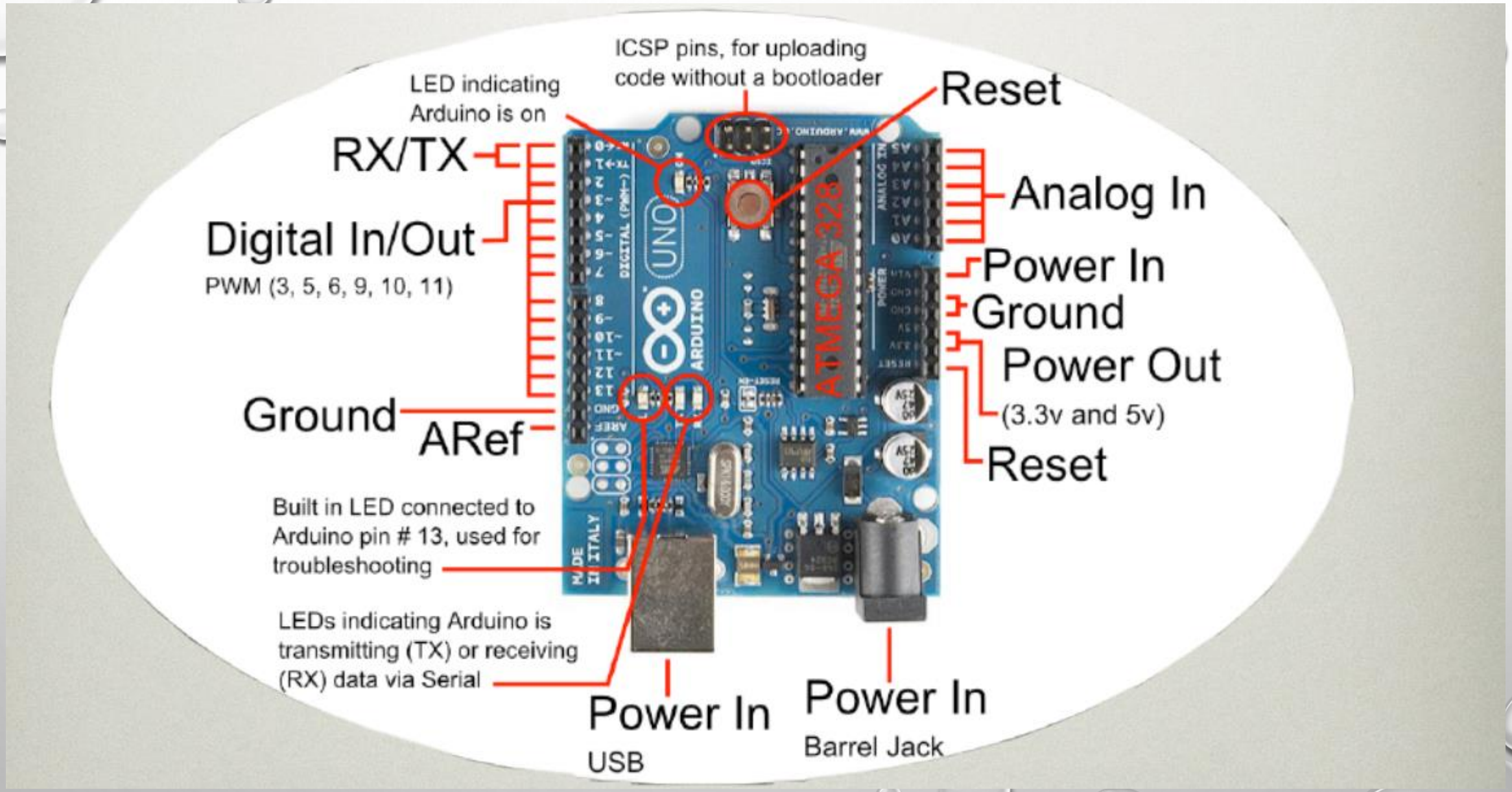




ARDUINO UNO / NANO



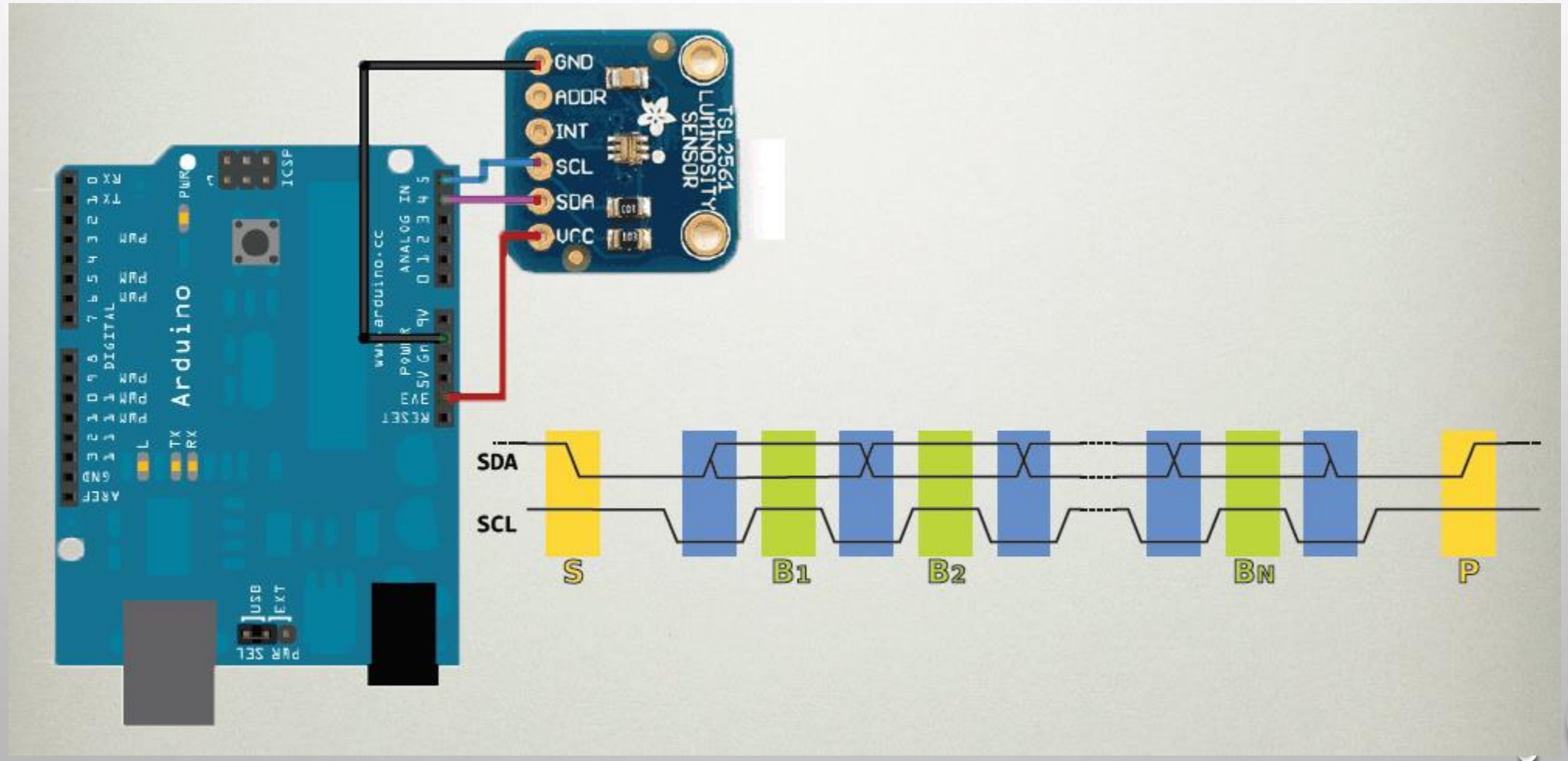



ICSP pins, for uploading code without a bootloader
LED indicating Arduino is on
Reset
RX/TX
Analog In
Digital In/Out
PWM (3, 5, 6, 9, 10, 11)
Power In
Ground
Power Out
(3.3v and 5v)
Ground
ARef
Reset
Built in LED connected to Arduino pin # 13, used for troubleshooting
LEDs indicating Arduino is transmitting (TX) or receiving (RX) data via Serial
Power In
USB
Power In
Barrel Jack

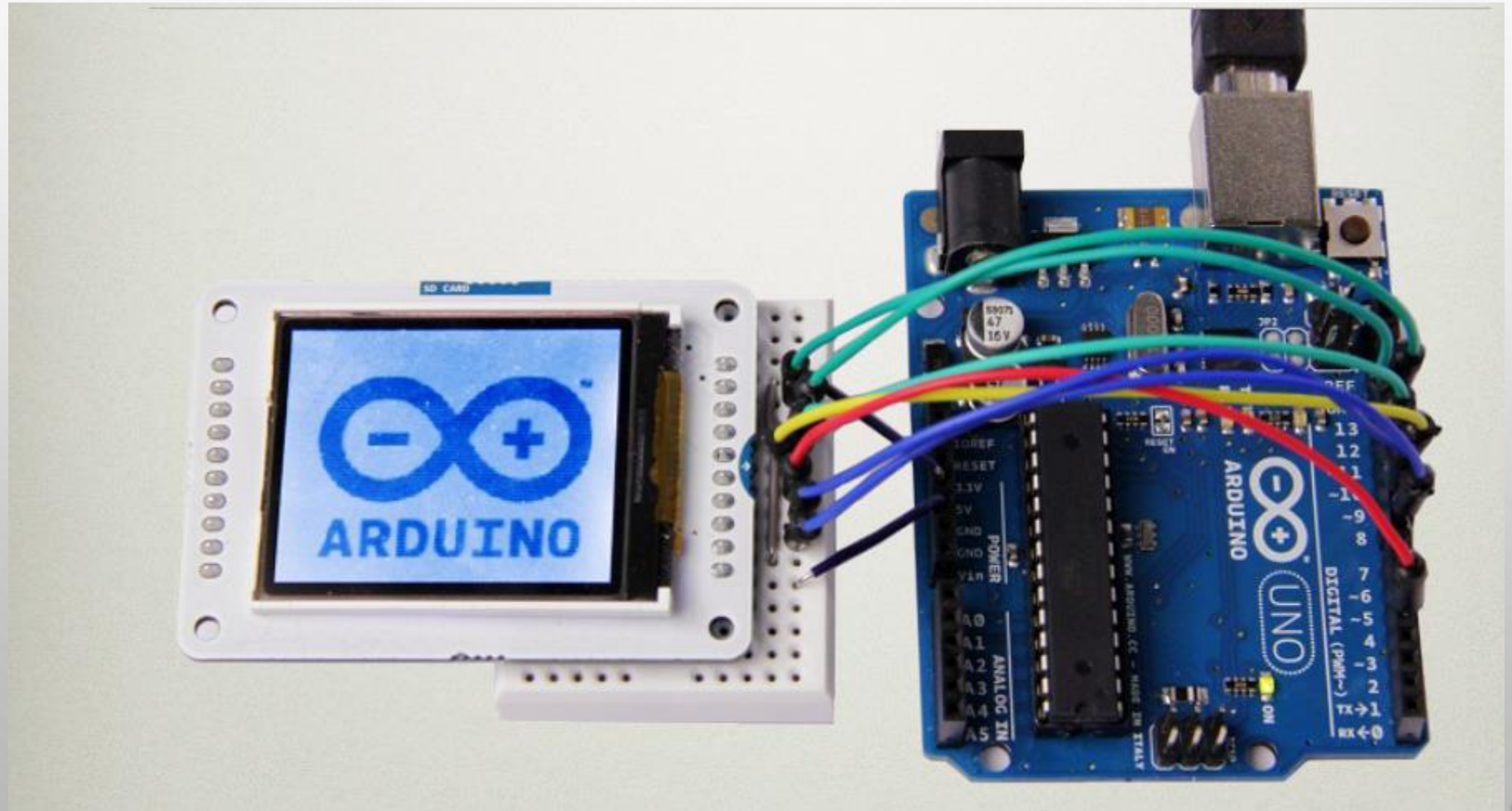
GENERAL PURPOSE INPUT/OUTPUT GPIO

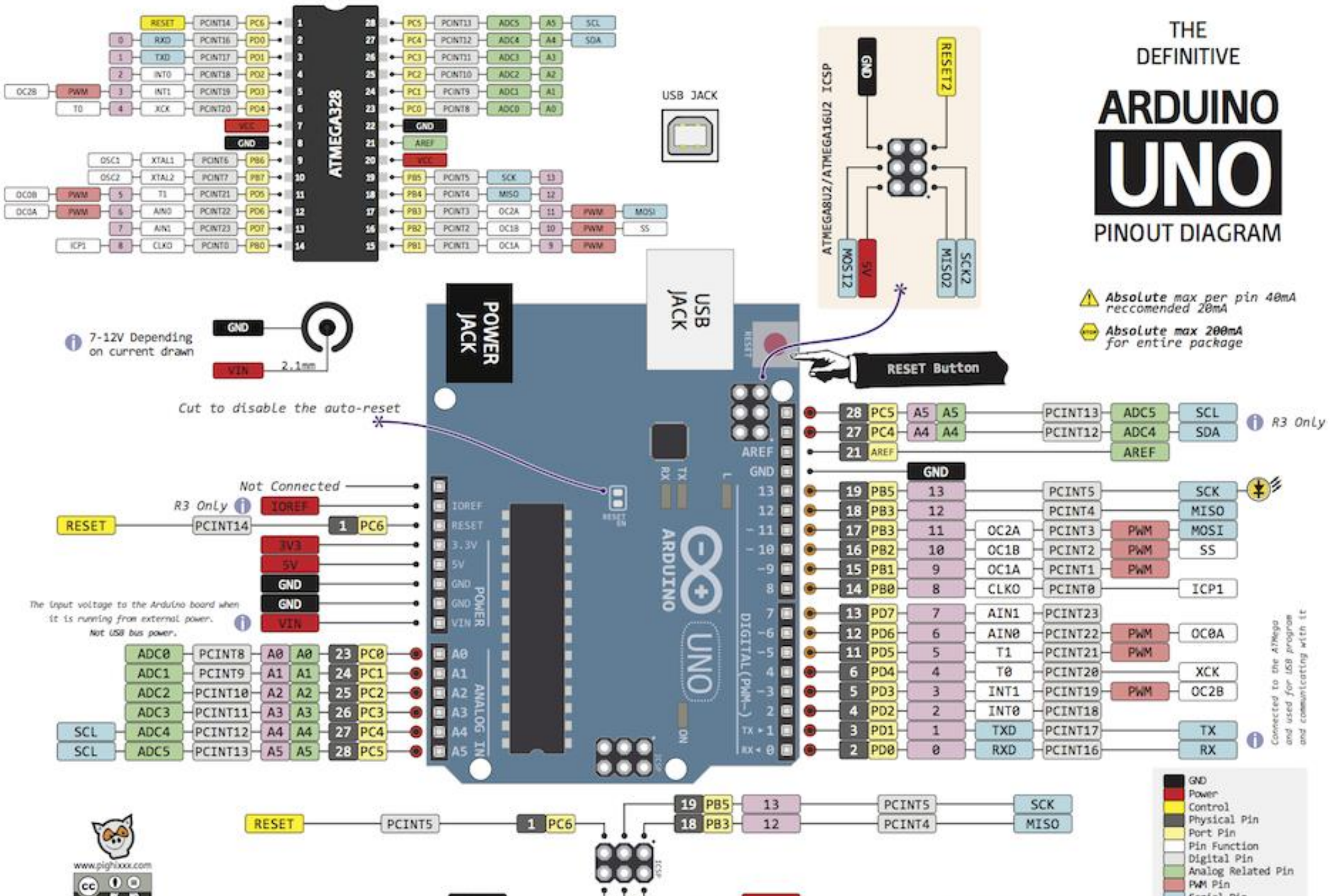
- CAN BE USED AS DIGITAL INPUT OR OUTPUT
- EACH PIN UNIQUELY ASSIGNABLE
- “ANALOG” (PWM OUTPUT ON DIGITAL PINS 3,5,6,9,10,11)
- ANALOG PINS 0-5 CAN ALSO BE USED AS GPIO

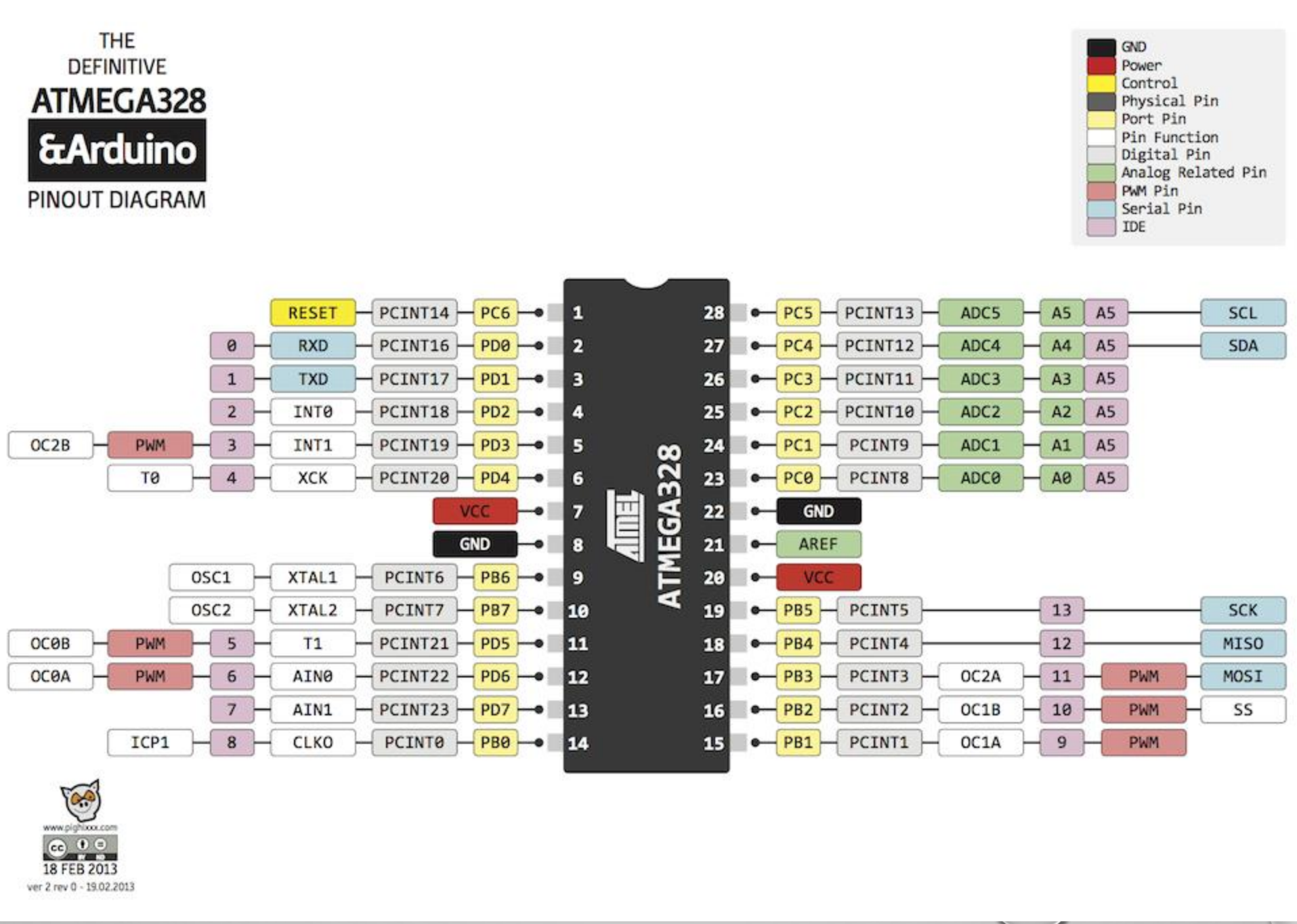
I2C / 2WI


GND
ADDR
INT
SCL
SDA
VCC
TSL2561
LUMINOSITY
SENSOR
Arduino
SDA
SCL
S
B1
B2
BN
P

SPI




THE
DEFINITIVE
ARDUINO
UNO
PINOUT DIAGRAM
ATMEGA328
USB JACK
ATMEGA8U2/ATMEGA16U2 ICSP
GND
RESET2
MOSI2
5V
MISO2
SCK2
Absolute max per pin 40mA
reccomended 20mA
Absolute max 200mA
for entire package
POWER JACK
USB JACK
7-12V Depending on current drawn
GND
VIN
2.1mm
RESET Button
Cut to disable the auto-reset
Not Connected
R3 Only
IOREF
RESET
PCINT14
1 PC6
3V3
5V
GND
GND
VIN
The input voltage to the Arduino board when it is running from external power.
Not USB bus power.
ADC0 PCINT8 A0 A0 23 PC0
ADC1 PCINT9 A1 A1 24 PC1
ADC2 PCINT10 A2 A2 25 PC2
ADC3 PCINT11 A3 A3 26 PC3
SCL ADC4 PCINT12 A4 A4 27 PC4
SCL ADC5 PCINT13 A5 A5 28 PC5
28 PC5 A5 A5 PCINT13 ADC5 SCL
27 PC4 A4 A4 PCINT12 ADC4 SDA
21 AREF AREF
GND
19 PB5 13 PCINT5 SCK
18 PB3 12 PCINT4 MISO
17 PB3 11 OC2A PCINT3 PWM MOSI
16 PB2 10 OC1B PCINT2 PWM SS
15 PB1 9 OC1A PCINT1 PWM
14 PB0 8 CLKO PCINT0 ICP1
13 PD7 7 AIN1 PCINT23
12 PD6 6 AIN0 PCINT22 PWM OC0A
11 PD5 5 T1 PCINT21 PWM
6 PD4 4 T0 PCINT20 XCK
5 PD3 3 INT1 PCINT19 PWM OC2B
4 PD2 2 INT0 PCINT18
3 PD1 1 TXD PCINT17 TX
2 PD0 0 RXD PCINT16 RX
R3 Only
Connected to the ATMega and used for USB program and communicating with it
RESET PCINT5 1 PC6
19 PB5 13 PCINT5 SCK
18 PB3 12 PCINT4 MISO
ICSP
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
www.pighixxx.com

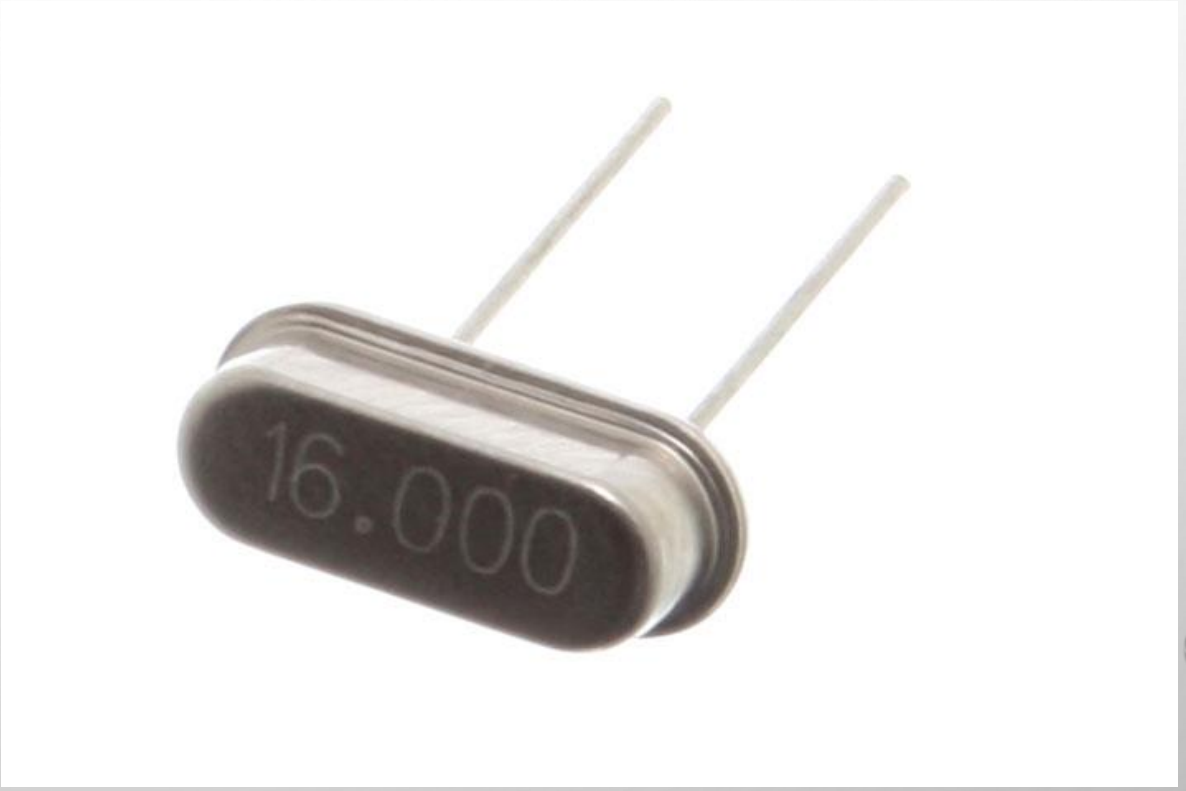

THE
DEFINITIVE
ATMEGA328
&Arduino
PINOUT DIAGRAM
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
ATMEGA328
RESET PCINT14 PC6 1
0 RXD PCINT16 PD0 2
1 TXD PCINT17 PD1 3
2 INT0 PCINT18 PD2 4
OC2B PWM 3 INT1 PCINT19 PD3 5
T0 4 XCK PCINT20 PD4 6
VCC 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 A5 SCL
27 PC4 PCINT12 ADC4 A4 A5 SDA
26 PC3 PCINT11 ADC3 A3 A5
25 PC2 PCINT10 ADC2 A2 A5
24 PC1 PCINT9 ADC1 A1 A5
23 PC0 PCINT8 ADC0 A0 A5
22 GND
21 AREF
20 VCC
19 PB5 PCINT5 13 SCK
18 PB4 PCINT4 12 MISO
17 PB3 PCINT3 OC2A 11 PWM MOSI
16 PB2 PCINT2 OC1B 10 PWM SS
15 PB1 PCINT1 OC1A 9 PWM
www.pighixxx.com
18 FEB 2013
ver 2 rev 0 - 19.02.2013

CONNECTING MICROCONTROLLERS TO OUTSIDE WORLD

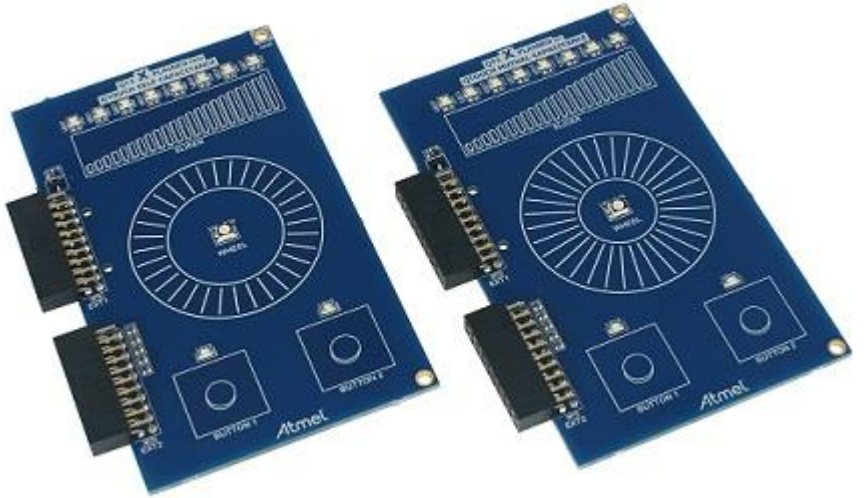
INTERFACING

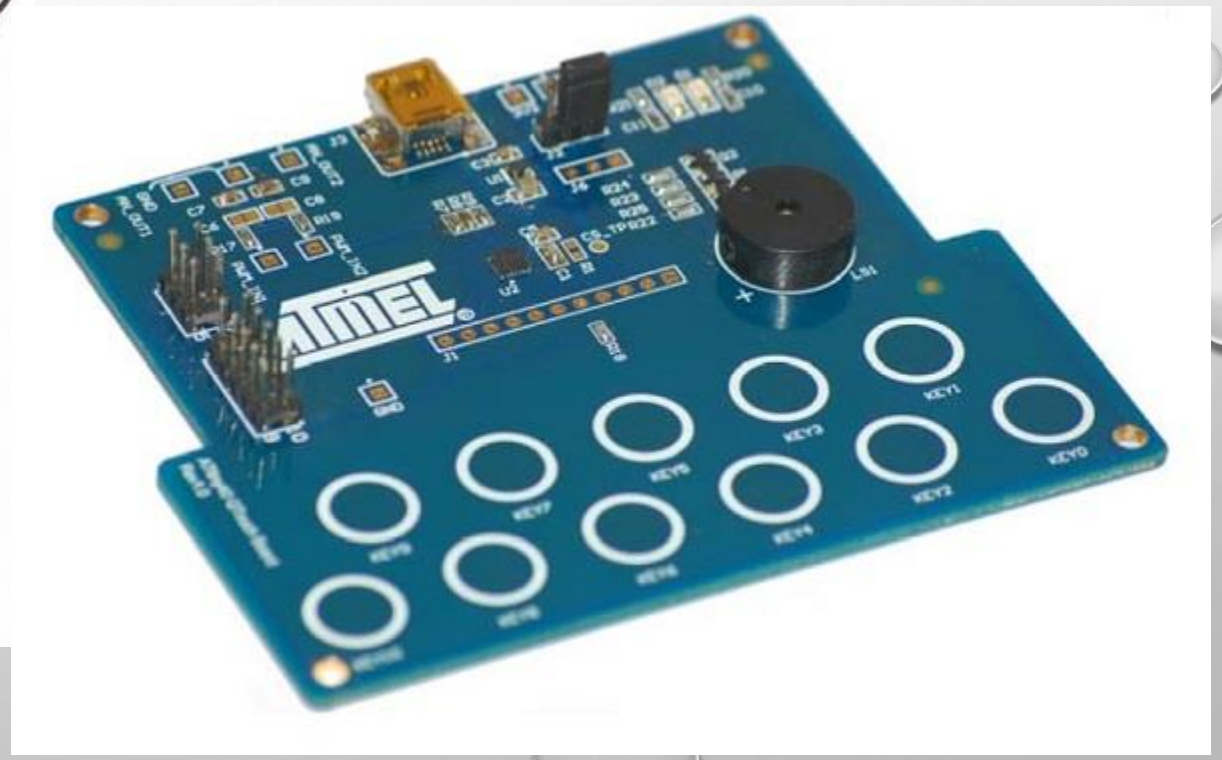
TYPES OF CONNECTIONS

- MICROCONTROLLER DRIVER
  - CLOCK (INTERNAL/EXTERNAL)
  - POWER SOURCE
- INPUTS (GPIOS/PROTOCOLS)
- OUTPUTS
- INTERNAL WIRING (TIMERS/COUNTERS)
- INTERNAL MEMORY



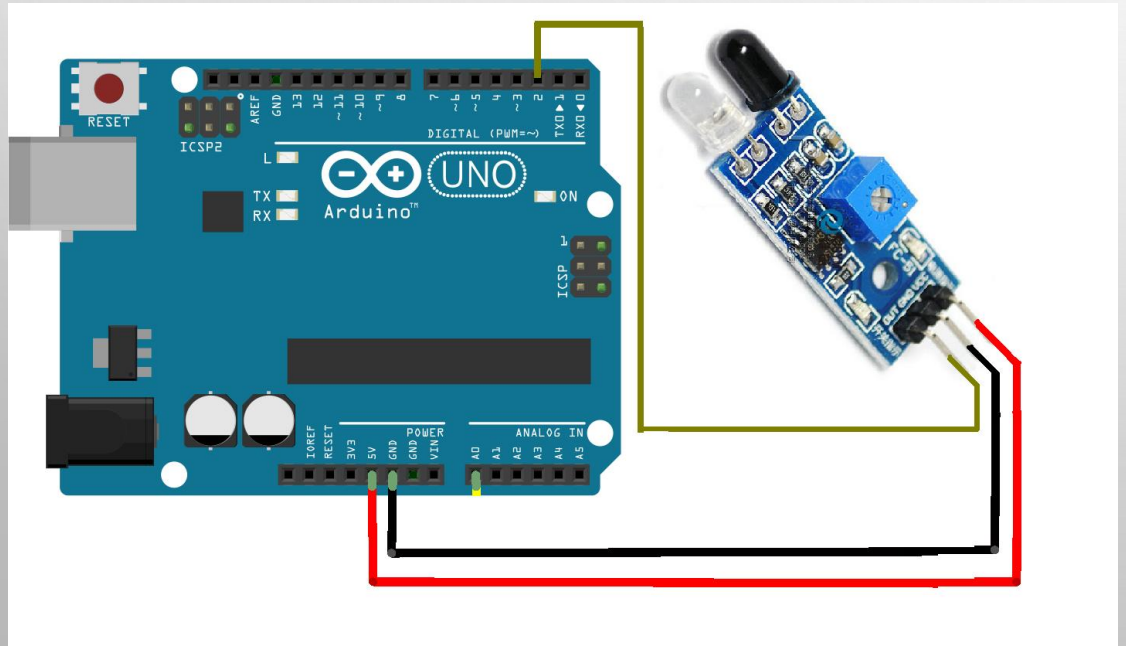

AVR
QTouch

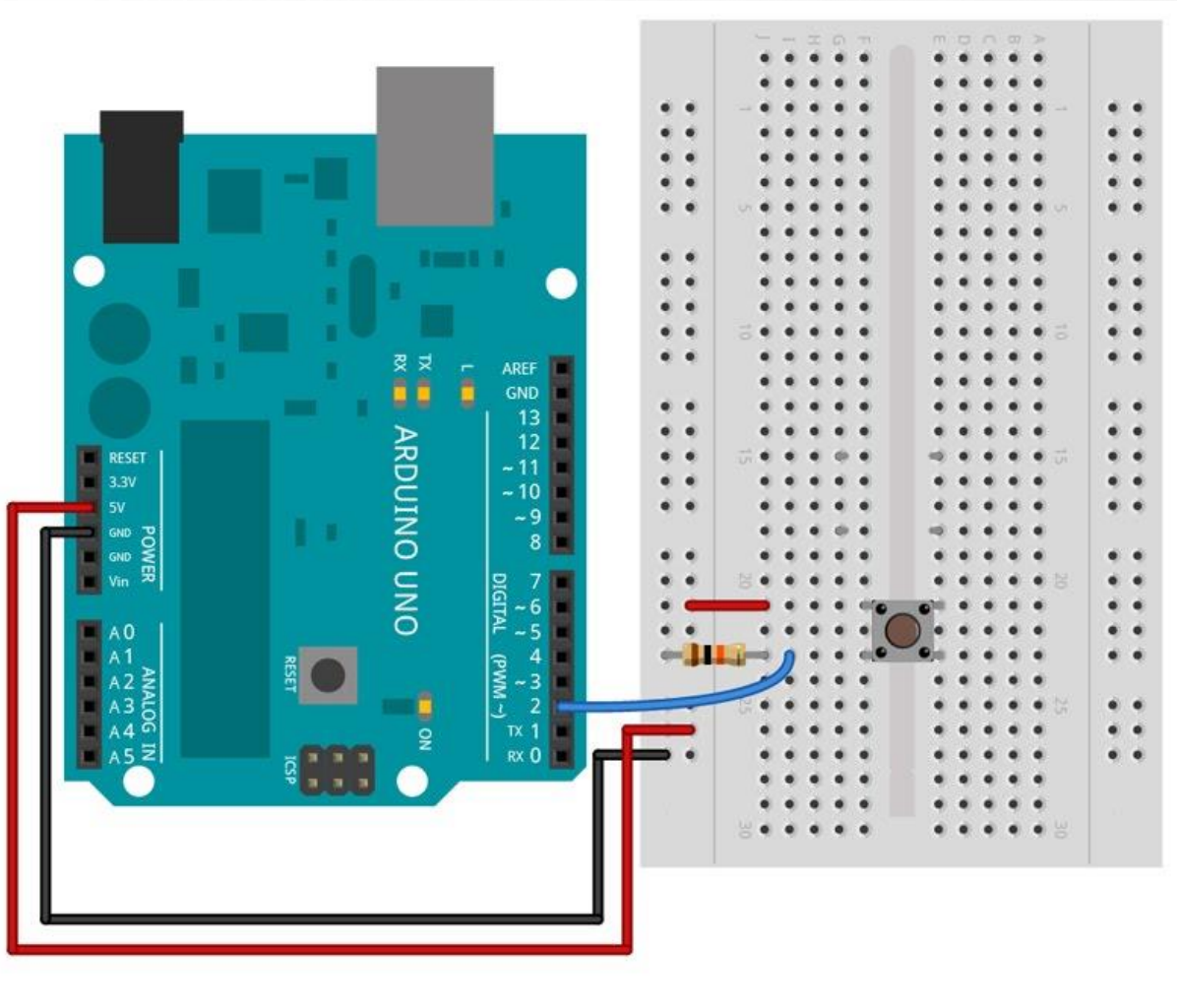

Atmel


ATMEL

DIGITAL INPUT

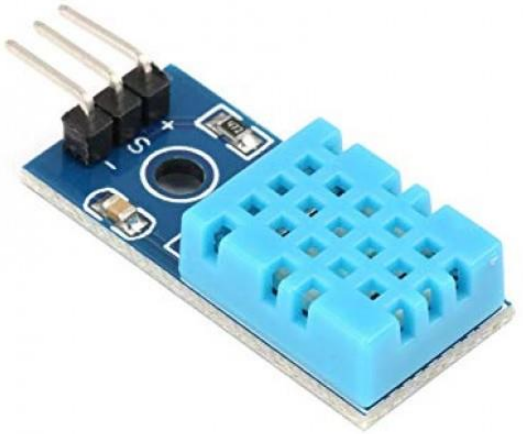
- BUTTON
- SWITCH
- IR SENSOR

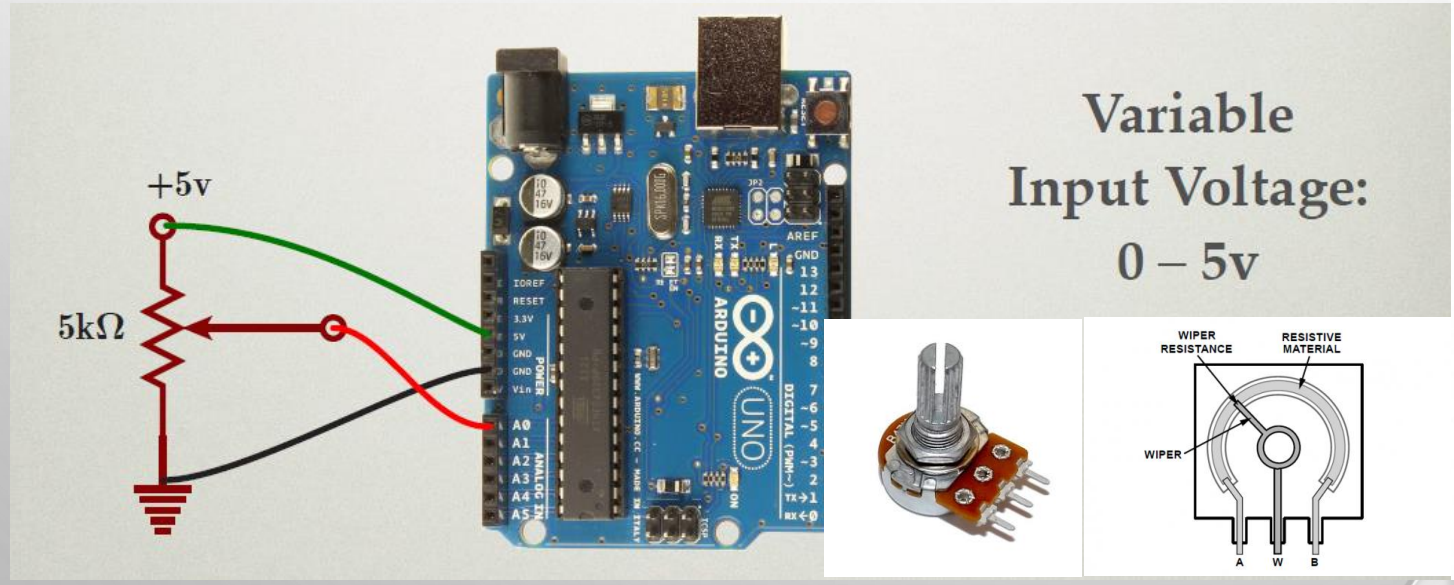


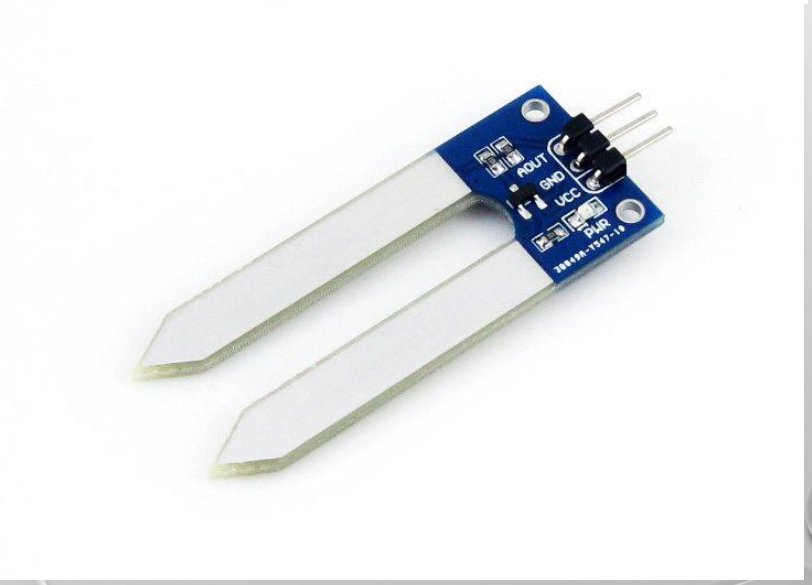


ANALOGUE INPUT

- HUMIDITY SENSOR
- TEMPERATURE SENSOR (LM35)

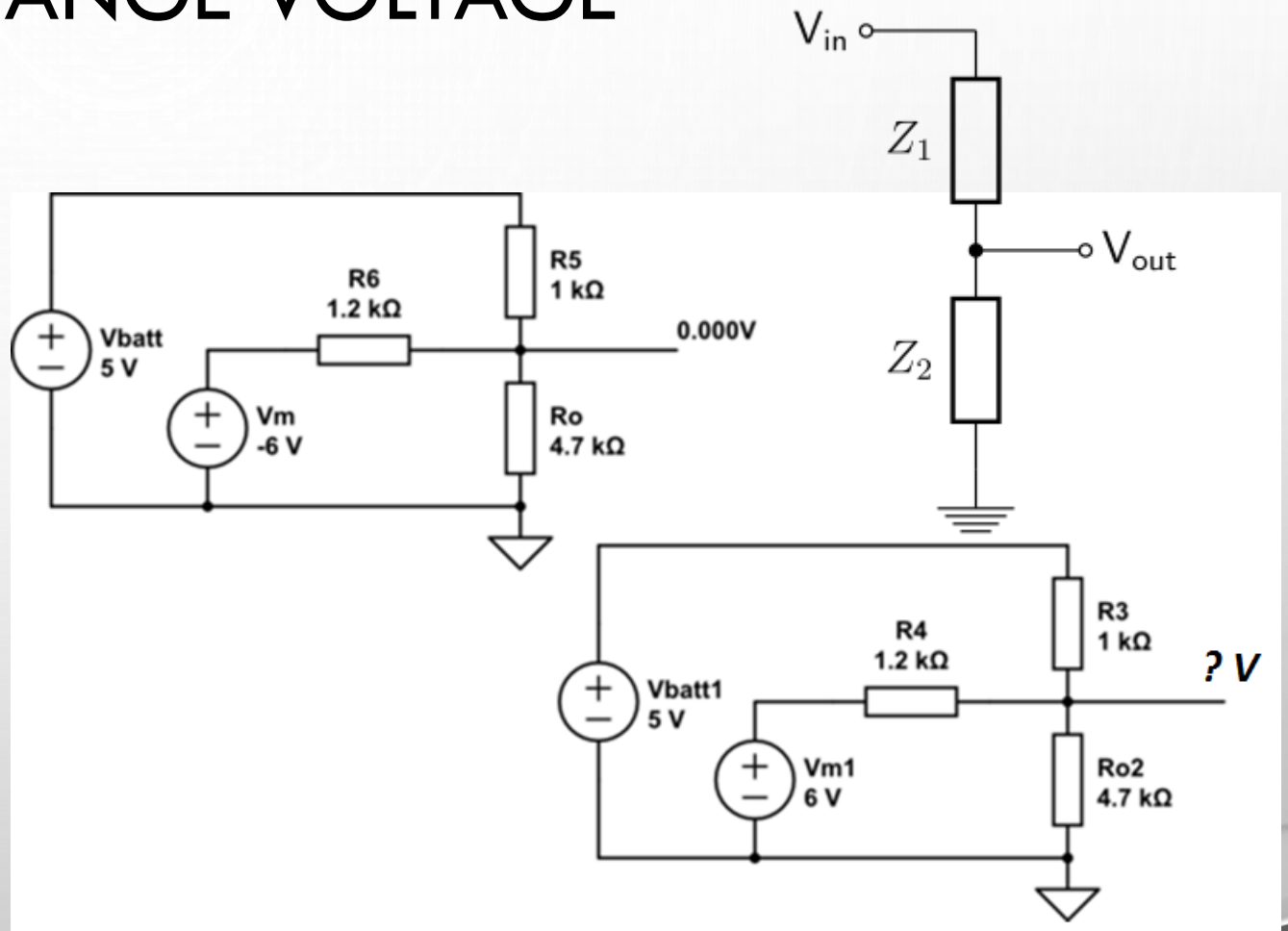






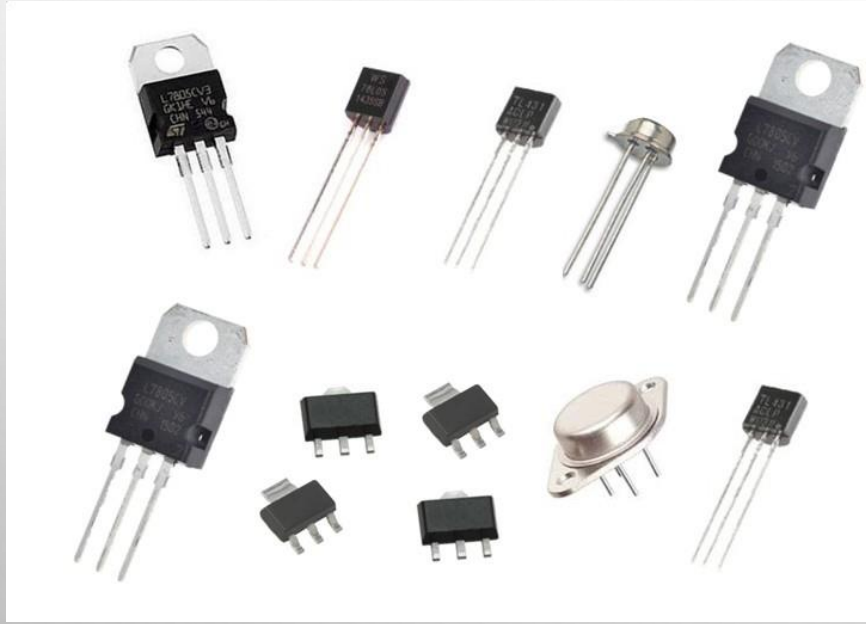
OUT OF RANGE VOLTAGE

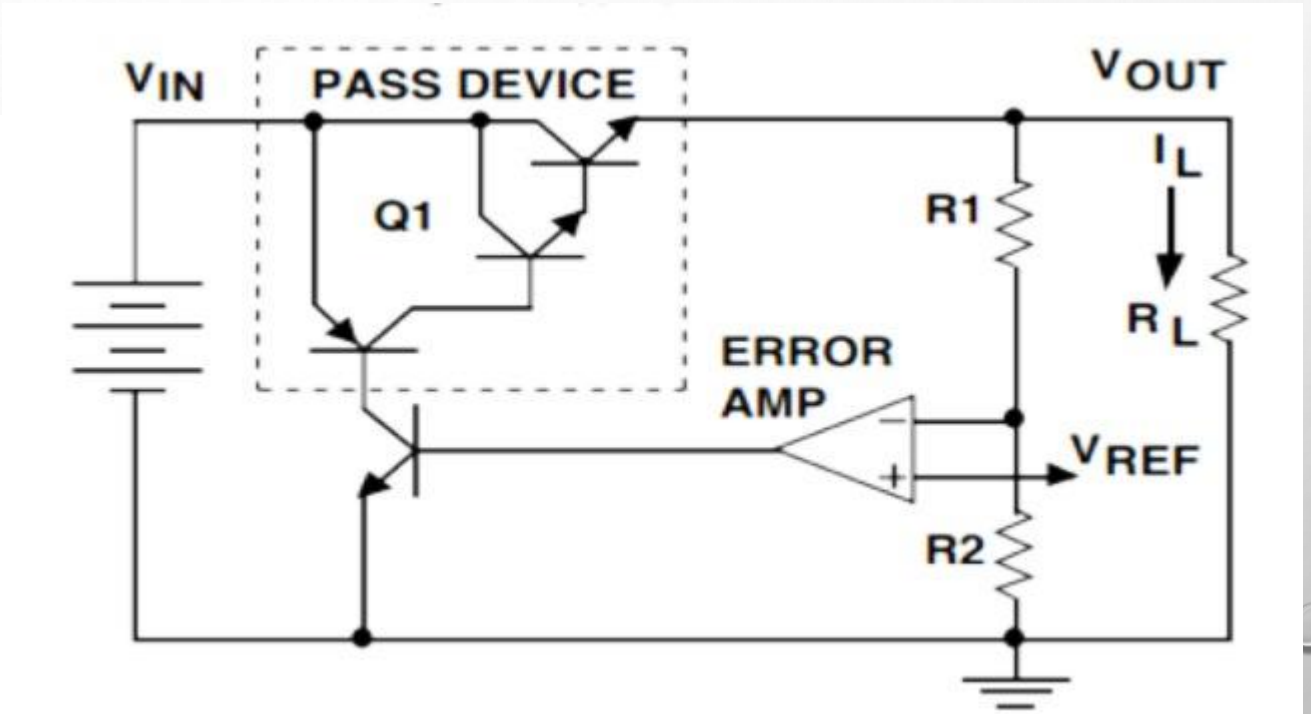
- **+6 TO -6 INPUT -> 0 TO 5 OUTPUT**
- **RESISTORS DROP VOLTAGE. WOULD IT NOT BE A SIMPLE FIX TO JUST USE RESISTORS TO DROP THE VOLTAGE ACCORDING TO OHMS LAW? BUT THEN, RESISTORS DROP VOLTAGE DEPENDING ON THE CURRENT FLOWING THROUGH THEM. THE MOMENT YOUR COMPONENT STARTS DRAWING LESS CURRENT, THE VOLTAGE SHOOTS UP AND KILLS IT.**
- **IS VOLTAGE DIVIDERS SAFE ?**



VOLTAGE REGULATOR

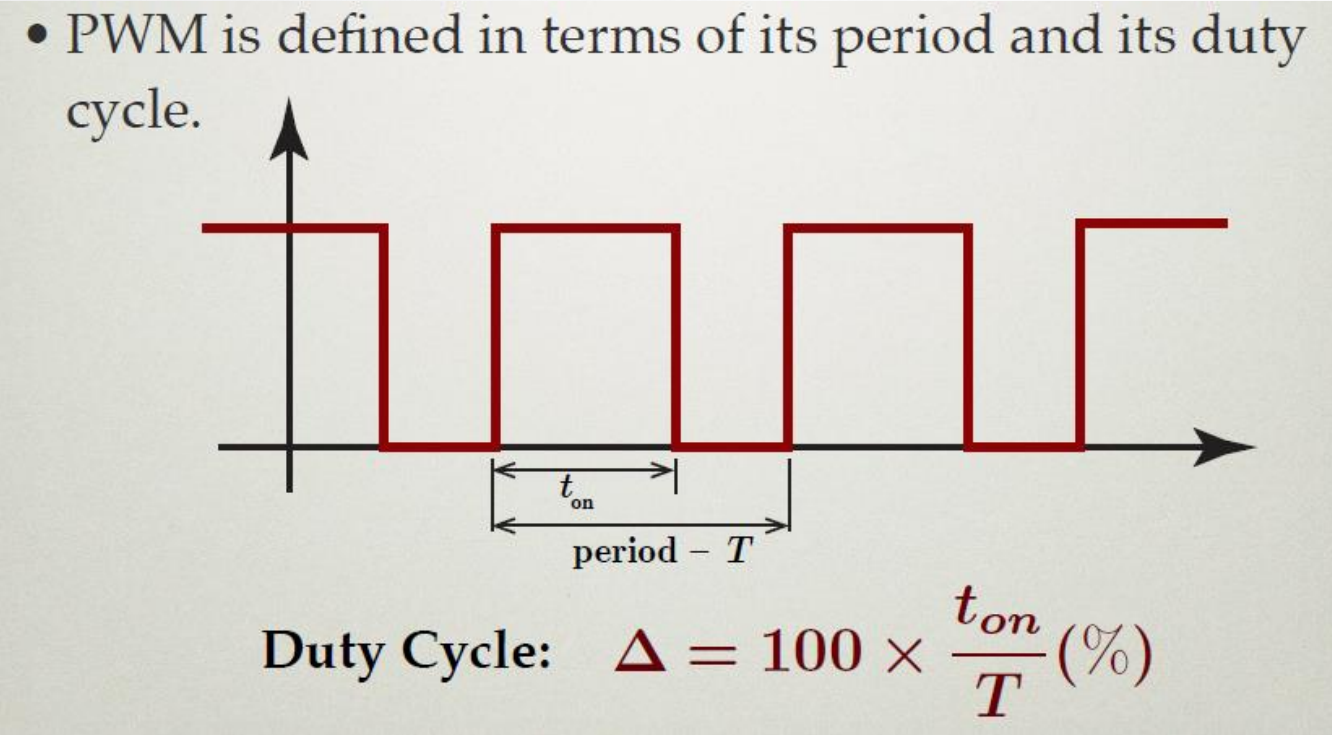
- 78XX SERIES

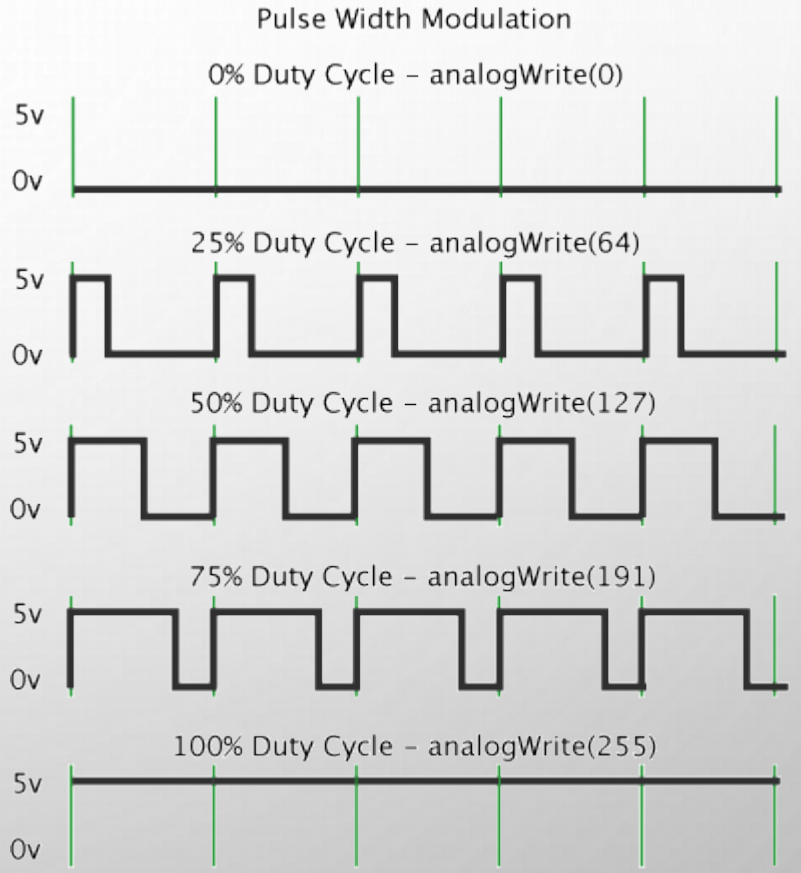




ANALOGUE OUTPUT EMULATION

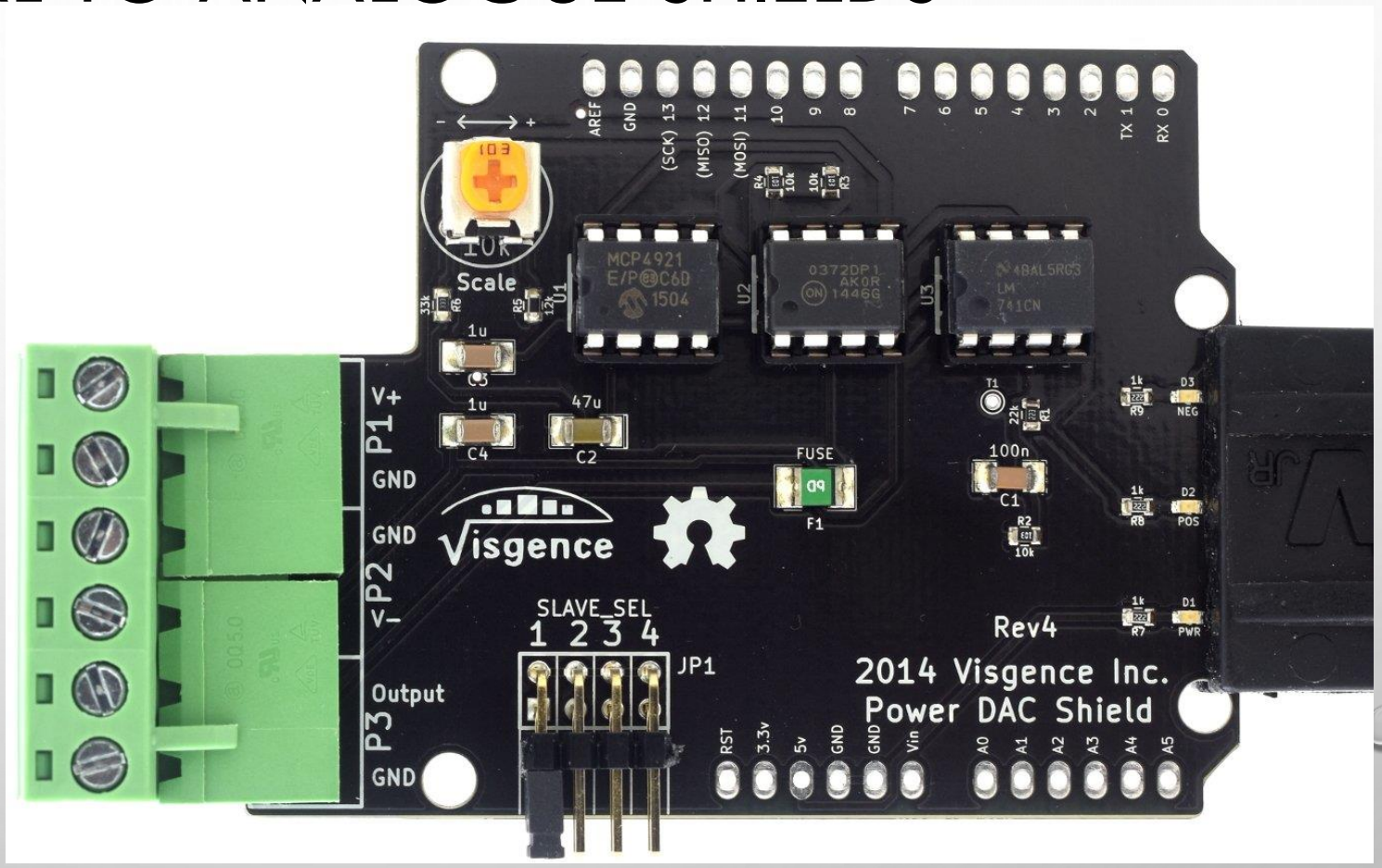
- PWM 500hz
  - analogWrite(pin, dutyCycle)



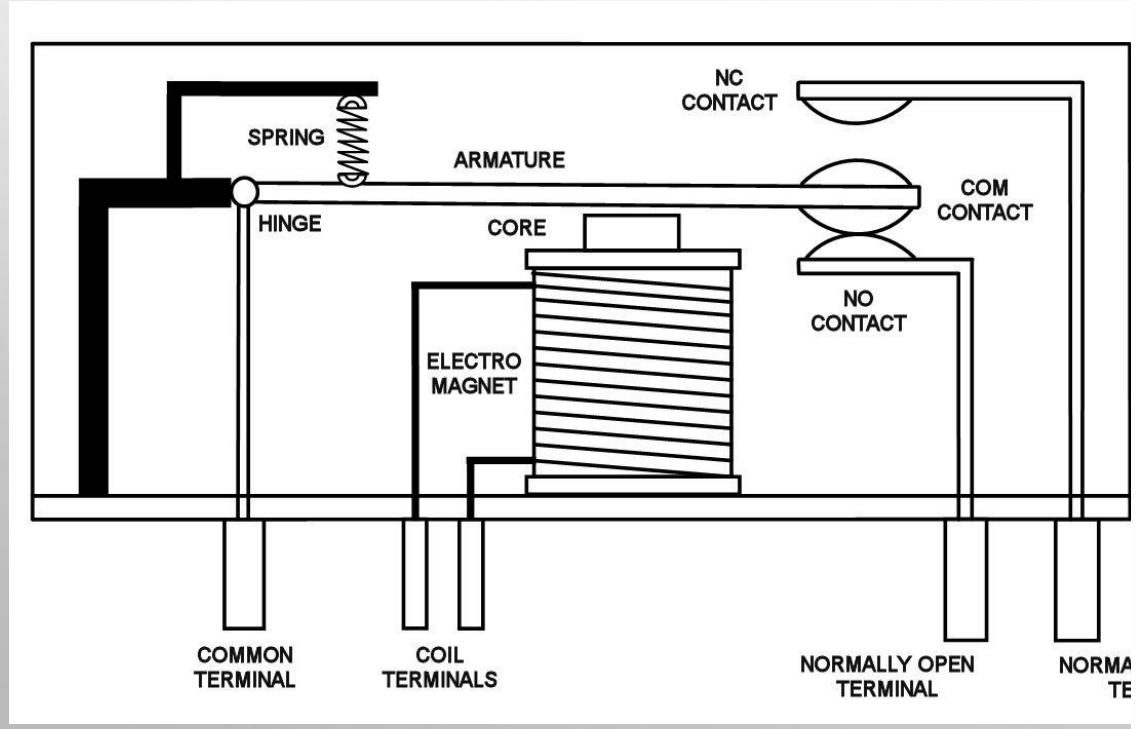


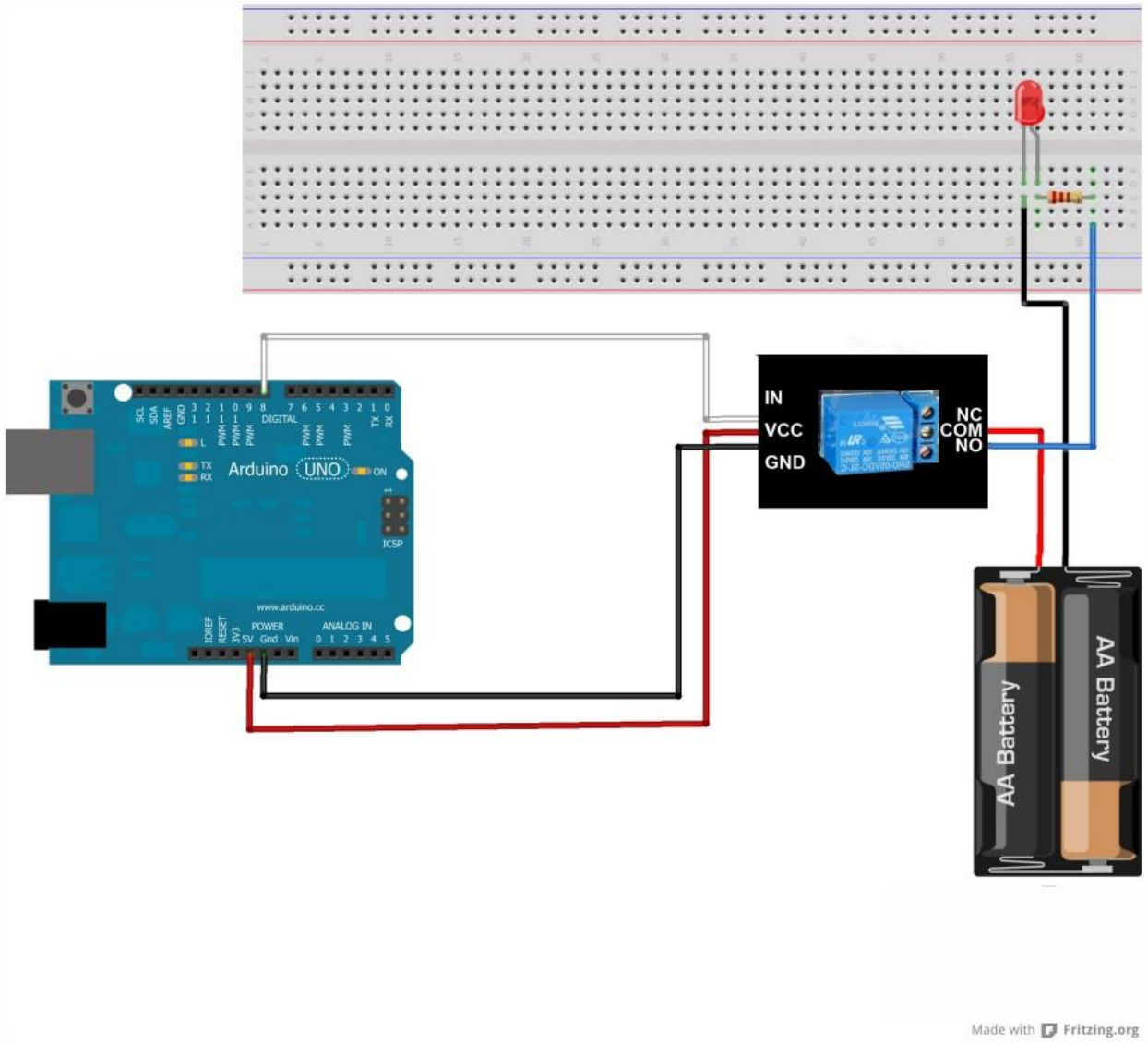
DIGITAL TO ANALOGUE SHIELDS

- TRUE AC WAVE FORM
- BIPOLAR POWER SUPPLY
- SPI OR I2C
- DC MOTORS



CIRCUI

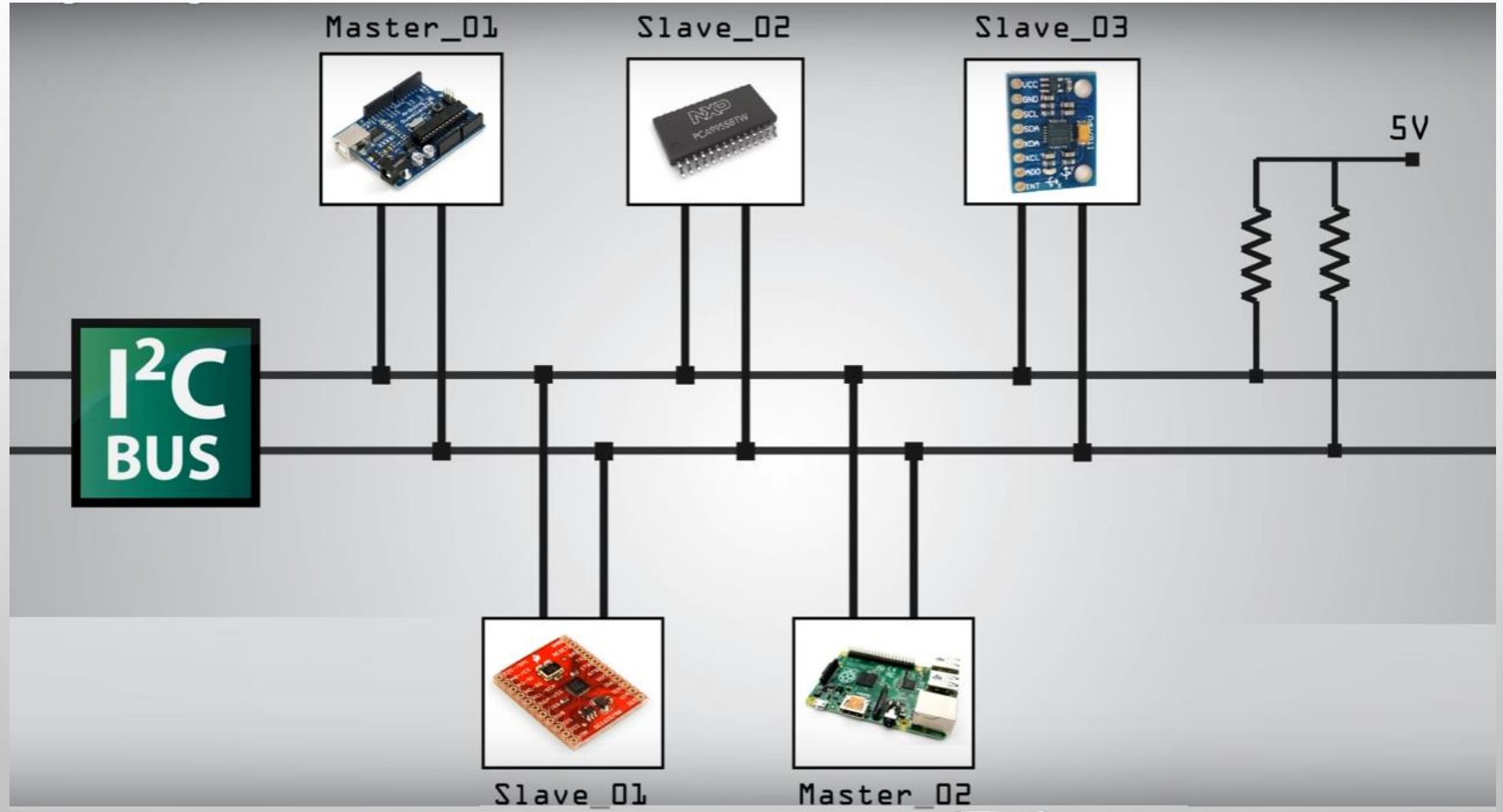

NC
CONTACT
SPRING
ARMATURE
COM
CONTACT
HINGE
CORE
NO
CONTACT
ELECTRO
MAGNET
COMMON
TERMINAL
COIL
TERMINALS
NORMALLY OPEN
TERMINAL
NORMA
TE


IN
VCC
GND
NC
COM
NO
Arduino
UNO
AA Battery
AA Battery
Made with Fritzing.org

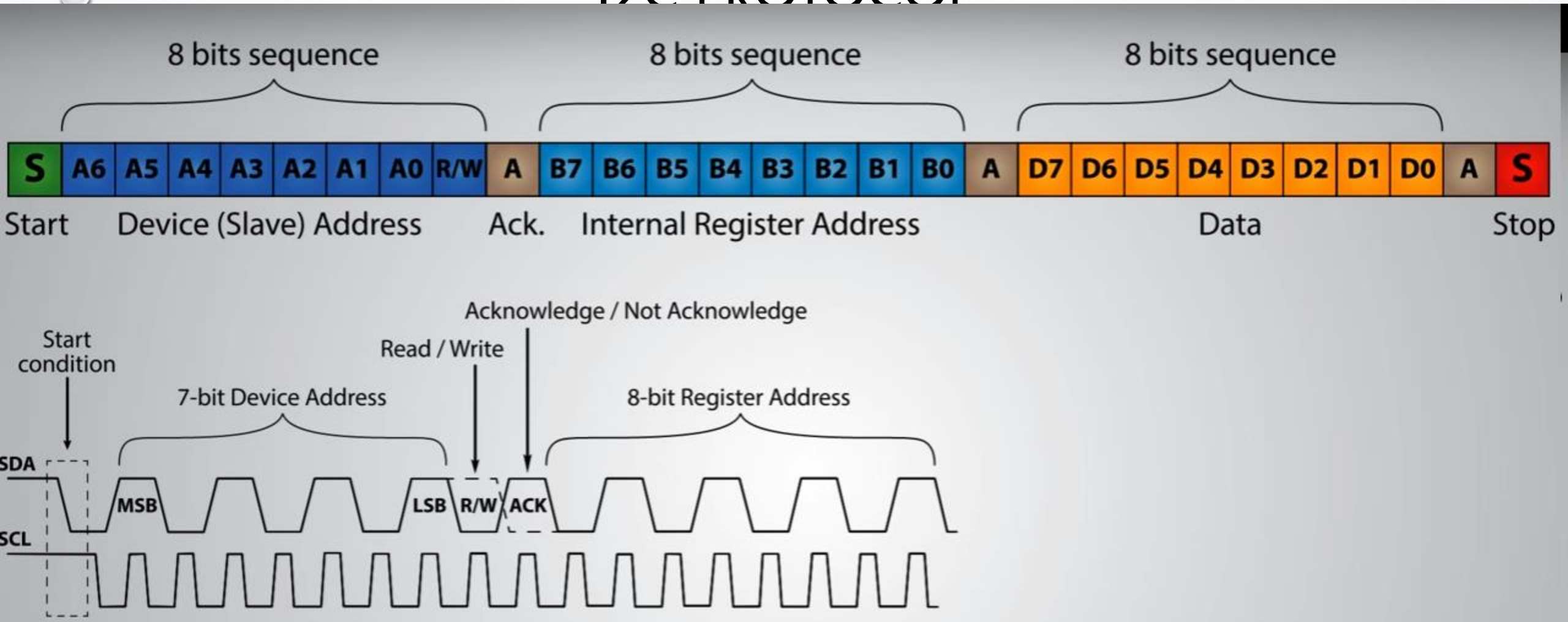
CONNECTING MICROCONTROLLERS TO OUTSIDE WORLD

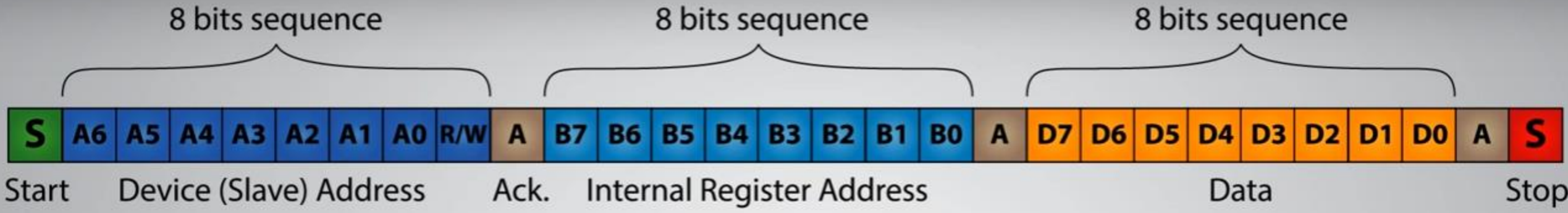
COMPLEX INTERFACES

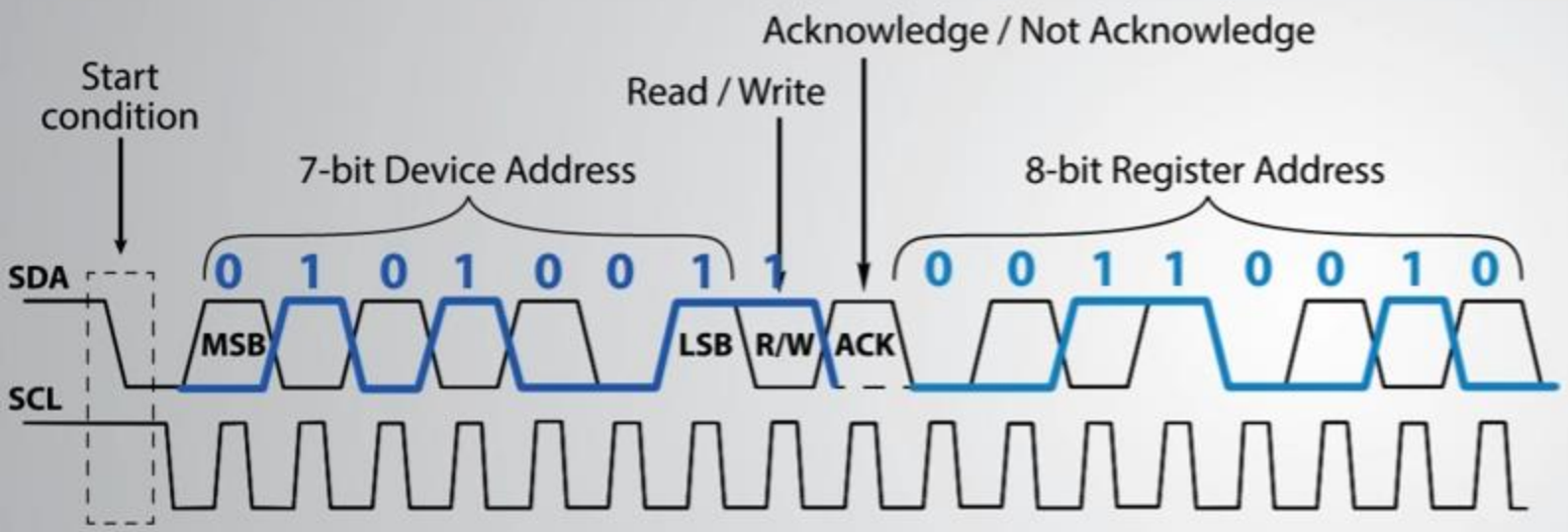
I2C



I2C PROTOCOL



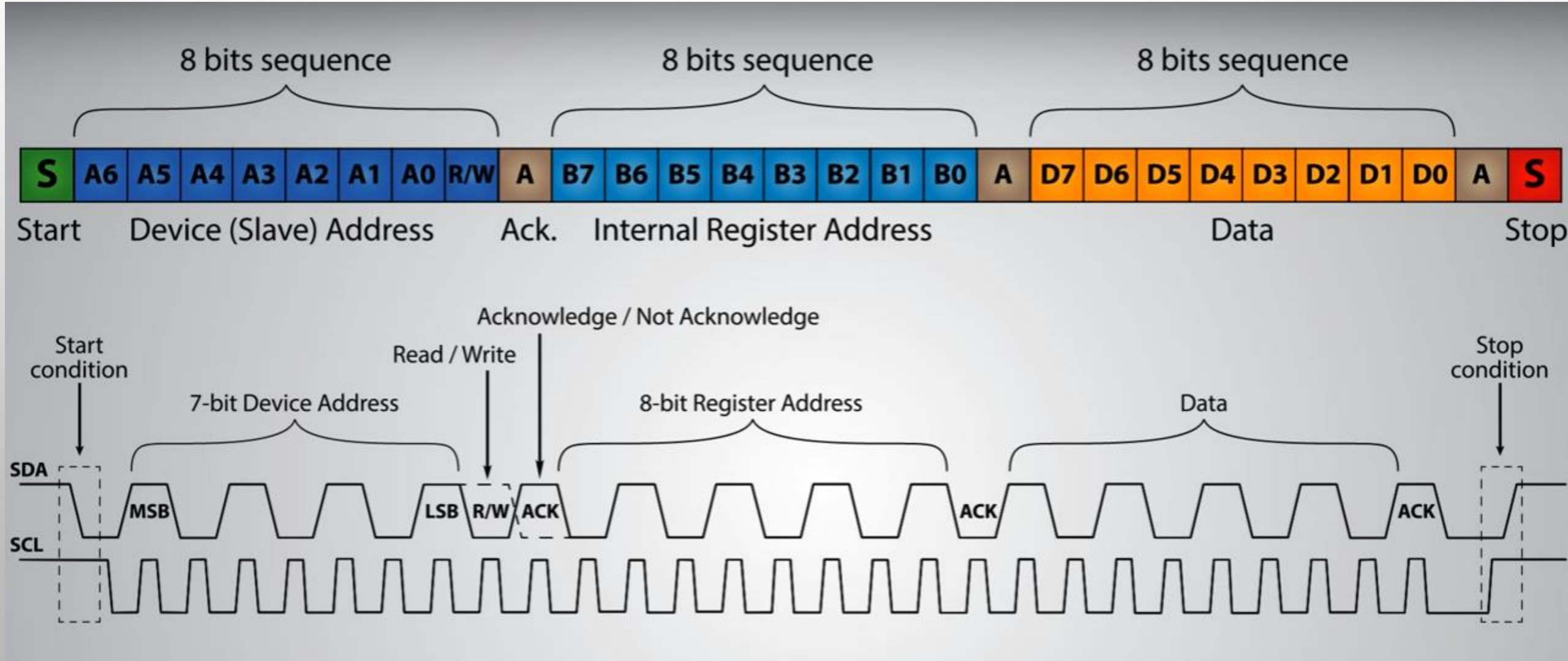




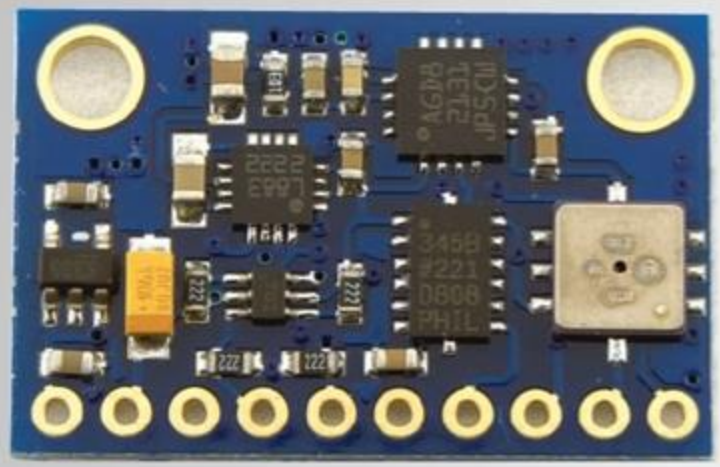
For example: ADXL345 Accelerometer

Device Address:_____0x53 or 0101 0011 (read mode, 8th bit high)
Internal Register Address
for the X Axis:_____0x32 or 0011 0010

I2C


8 bits sequence
8 bits sequence
8 bits sequence
S
A6
A5
A4
A3
A2
A1
A0
R/W
A
B7
B6
B5
B4
B3
B2
B1
B0
A
D7
D6
D5
D4
D3
D2
D1
D0
A
S
Start
Device (Slave) Address
Ack.
Internal Register Address
Data
Stop
Acknowledge / Not Acknowledge
Start condition
Read / Write
7-bit Device Address
8-bit Register Address
Data
Stop condition
SDA
SCL
MSB
LSB
R/W
ACK
ACK
ACK

GY - 80

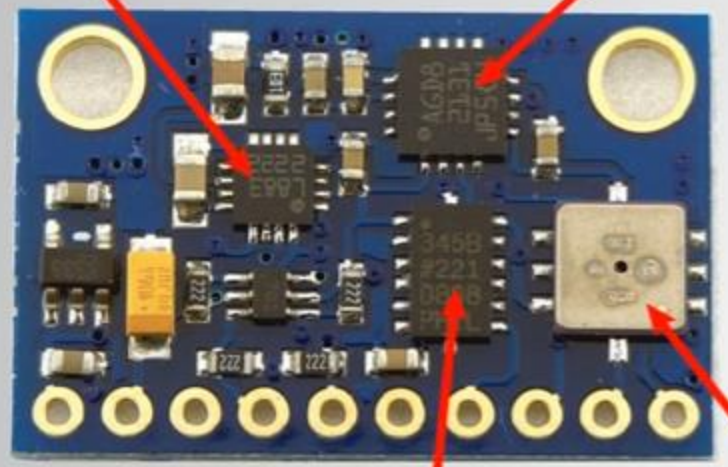


GY - 521

GY - 80

MC5883L
Magnetometer
Address: 0x1E

L3G4200D
Gyroscope
Address: 0x69

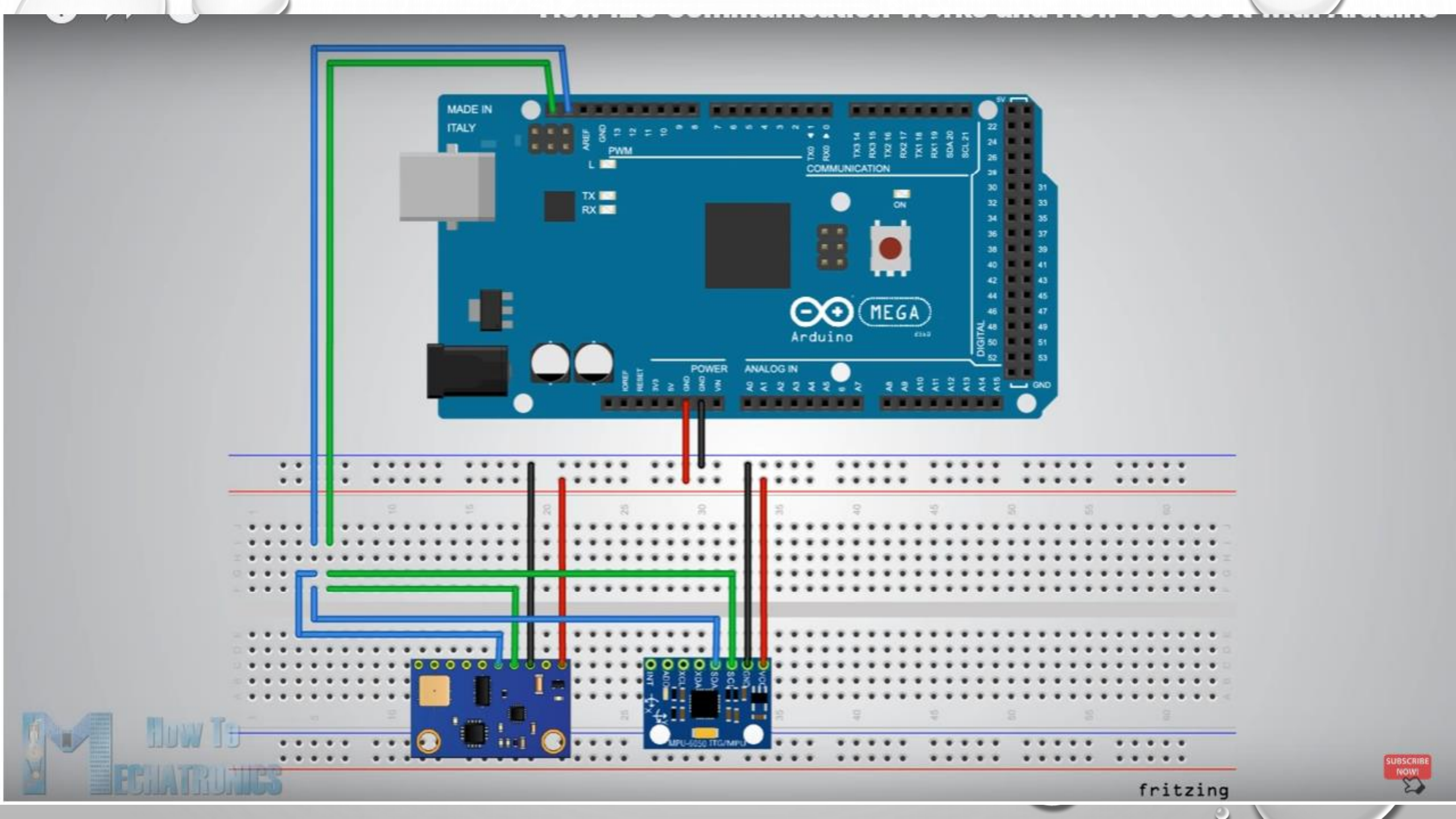


ADXL345
Accelerometer
Address: 0x53

BMP085
Barometer + Thermometer
Address: 0x77

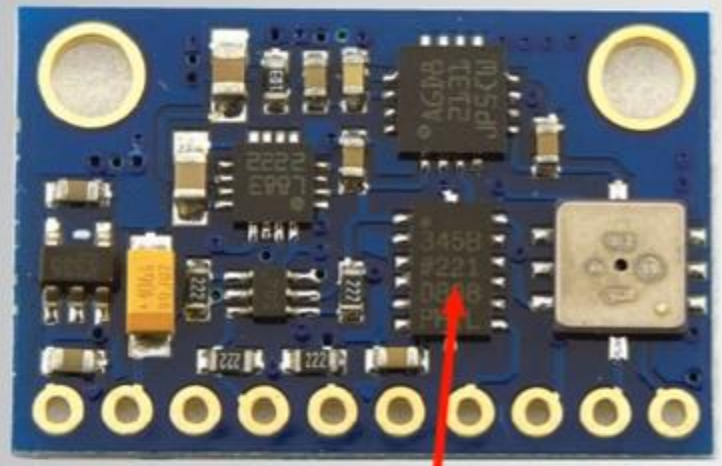
GY - 521

Note: These addresses are for Read Mode, included logic high at R/W bit.


MADE IN
ITALY
PWM
COMMUNICATION
POWER
ANALOG IN
DIGITAL
MEGA
Arduino
ON
TX
RX
L
GND
MPU-6050 TTG/MPU
fritzing

For Example:

Read data from the X Axis of the ADXL345 Accelerometer



ADXL345 Accelerometer Address: 0x53

Data Sheet **ADXL345**

REGISTER MAP

Table 19.

Address					
Hex	Dec	Name	Type	Reset Value	Description
0x00	0	DEVID	R	11100101	Device ID
0x01 to 0x1C	1 to 28	Reserved			Reserved; do not access
0x1D	29	THRESH_TAP	R/W	00000000	Tap threshold
0x1E	30	OFSX	R/W	00000000	X-axis offset
0x1F	31	OFSY	R/W	00000000	Y-axis offset
0x20	32	OFSZ	R/W	00000000	Z-axis offset
0x21	33	DUR	R/W	00000000	Tap duration
0x22	34	Latent	R/W	00000000	Tap latency
0x23	35	Window	R/W	00000000	Tap window
0x24	36	THRESH_ACT	R/W	00000000	Activity threshold
0x25	37	THRESH_INACT	R/W	00000000	Inactivity threshold
0x26	38	TIME_INACT	R/W	00000000	Inactivity time
0x27	39	ACT_INACT_CTL	R/W	00000000	Axis enable control for activity and inactivity detection
0x28	40	THRESH_FF	R/W	00000000	Free-fall threshold
0x29	41	TIME_FF	R/W	00000000	Free-fall time
0x2A	42	TAP_AXES	R/W	00000000	Axis control for single tap/double tap
0x2B	43	ACT_TAP_STATUS	R	00000000	Source of single tap/double tap
0x2C	44	BW_RATE	R/W	00001010	Data rate and power mode control
0x2D	45	POWER_CTL	R/W	00000000	Power-saving features control
0x2E	46	INT_ENABLE	R/W	00000000	Interrupt enable control
0x2F	47	INT_MAP	R/W	00000000	Interrupt mapping control
0x30	48	INT_SOURCE	R	00000010	Source of interrupts
0x31	49	DATA_FORMAT	R/W	00000000	Data format control
0x32	50	DATAX0	R	00000000	X-Axis Data 0
0x33	51	DATAX1	R	00000000	X-Axis Data 1
0x34	52	DATAY0	R	00000000	Y-Axis Data 0
0x35	53	DATAY1	R	00000000	Y-Axis Data 1
0x36	54	DATAZ0	R	00000000	Z-Axis Data 0
0x37	55	DATAZ1	R	00000000	Z-Axis Data 1
0x38	56	FIFO_CTL	R/W	00000000	FIFO control
0x39	57	FIFO_STATUS	R	00000000	FIFO status

*Figure from the datasheet of the ADXL345 Accelerometer

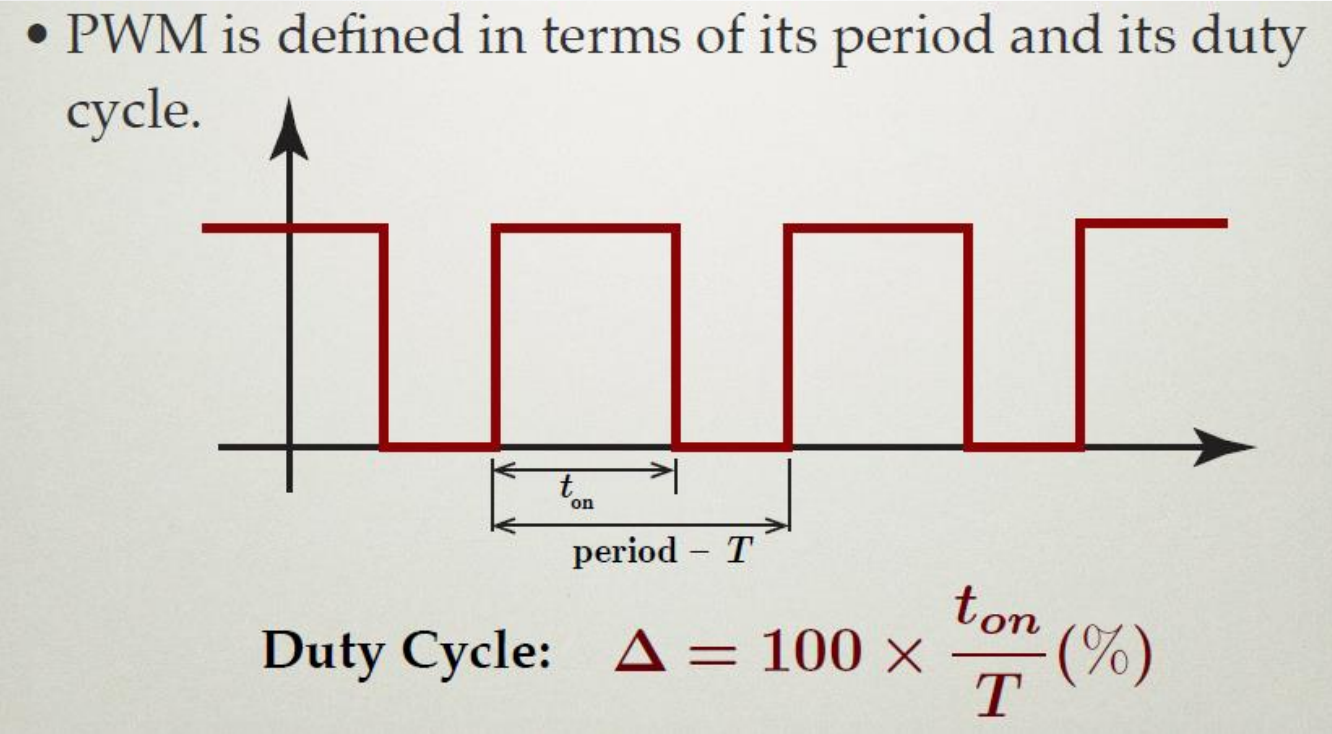
CONNECTION CONSIDERATIONS

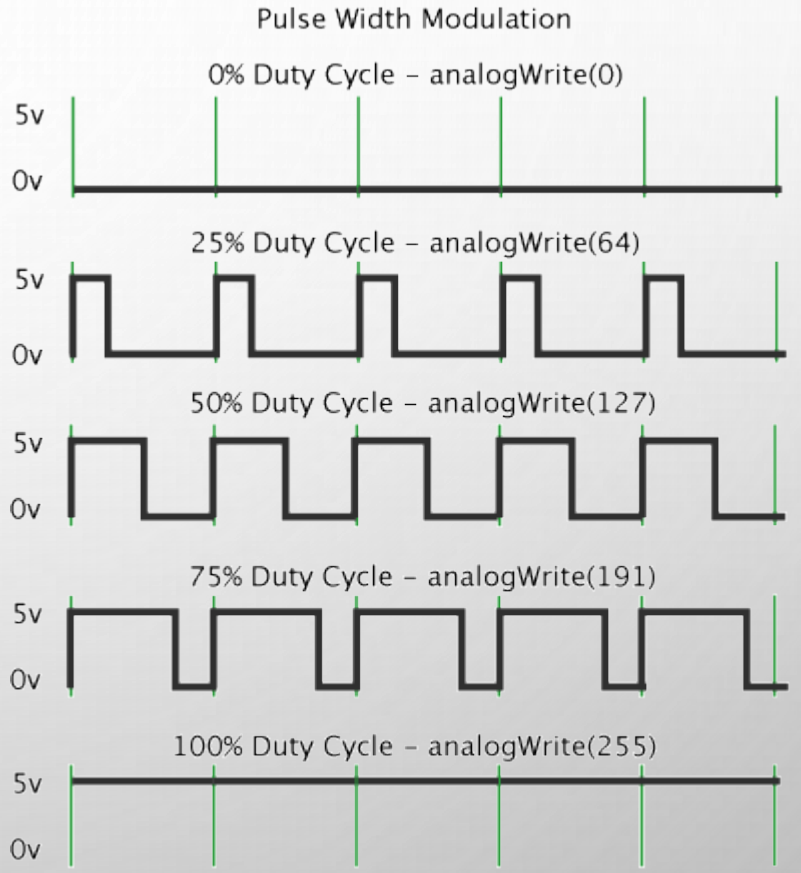
- SENSORS
  - VOLTAGE INPUT CONDITIONING +VE 0 TO 5 IS OK
  - WHAT IF SENSING IS REQUIRED IN –VE RANGE OR 10 V

- POWER
  - DIGITAL OUTPUT IS OK .
  - WHAT IF ANALOG OUTPUT REQUIRED ?
  - HIGHER DRIVE CURRENT OR AC ?

CONSIDERATIONS ON ARDUINO

- PWM 500hz
  - analogWrite(pin, dutyCycle)





PWM BIT TAGGING

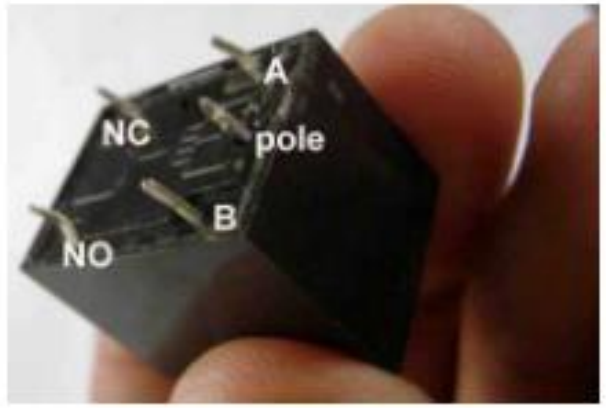
- VOID **SETUP**()
  {
  PINMODE(13, OUTPUT);
  }

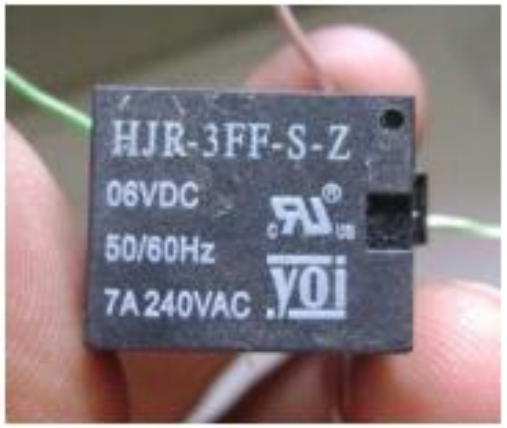
  VOID **LOOP**()
  {
  DIGITALWRITE(13, HIGH);
  DELAYMICROSECONDS(100); *// APPROXIMATELY 10% DUTY CYCLE @ 1KHZ*
  DIGITALWRITE(13, LOW);
  DELAYMICROSECONDS(1000 - 100);
  }

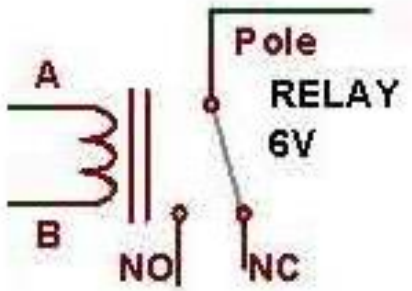
USING RELAYS

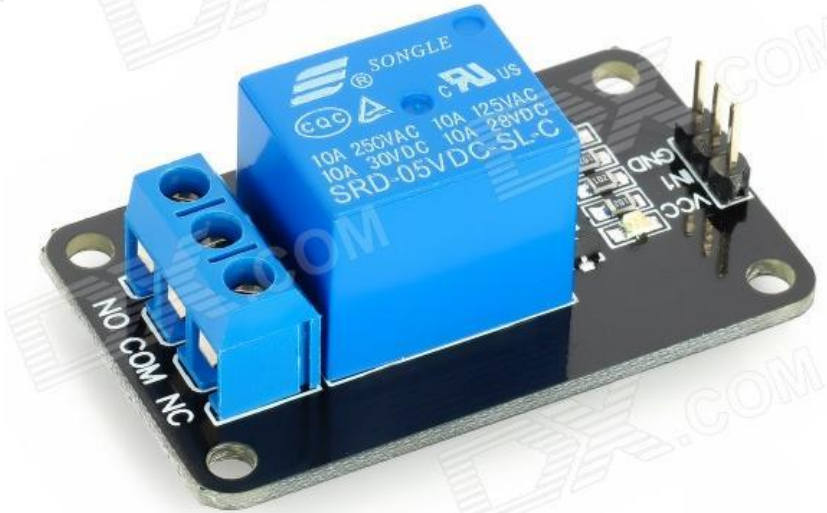
- [HTTP://WWW.BUILDCIRCUIT.COM/HOW-TO-USE-A-RELAY/](http://www.buildcircuit.com/how-to-use-a-relay/)

A relay shown in the picture is an electromagnetic or mechanical relay.

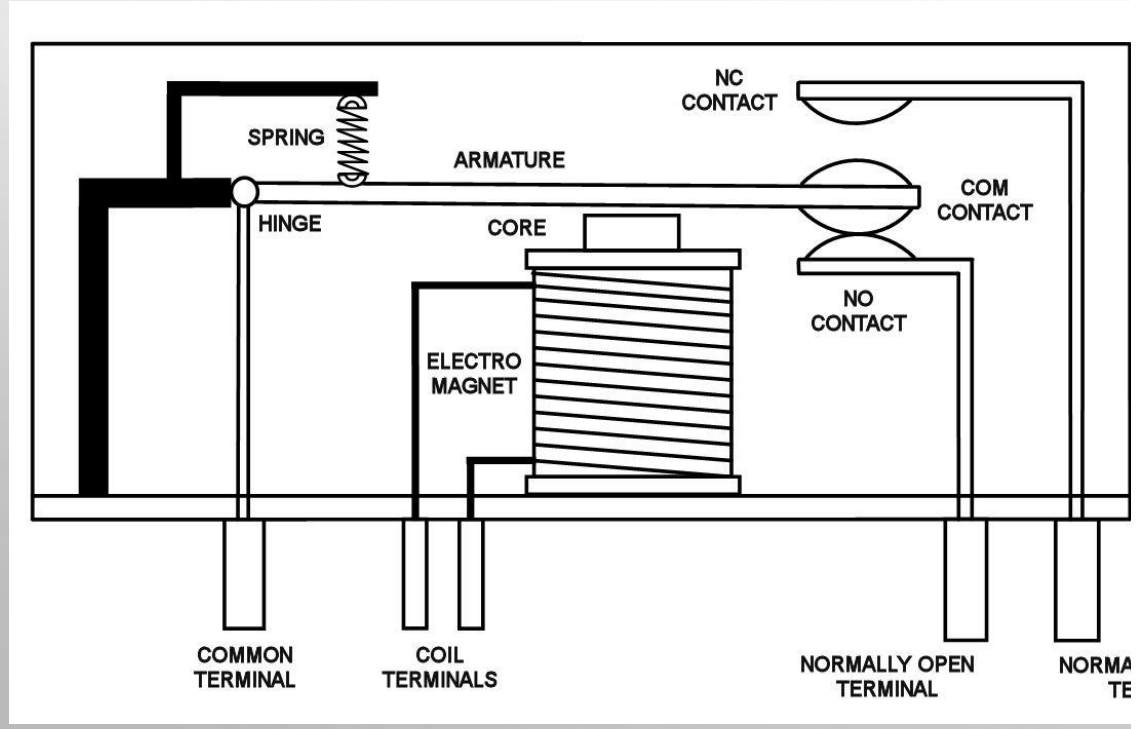


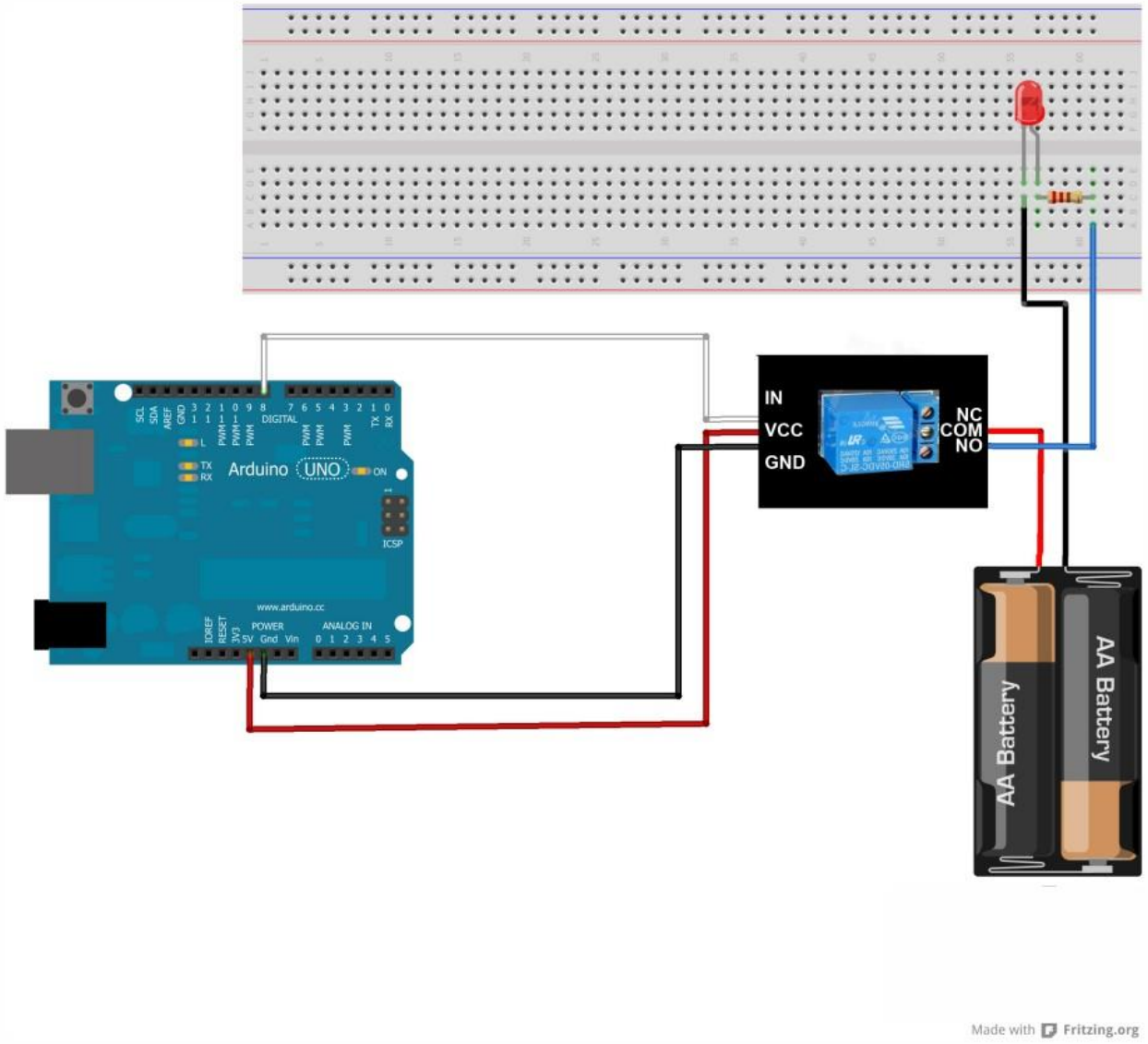






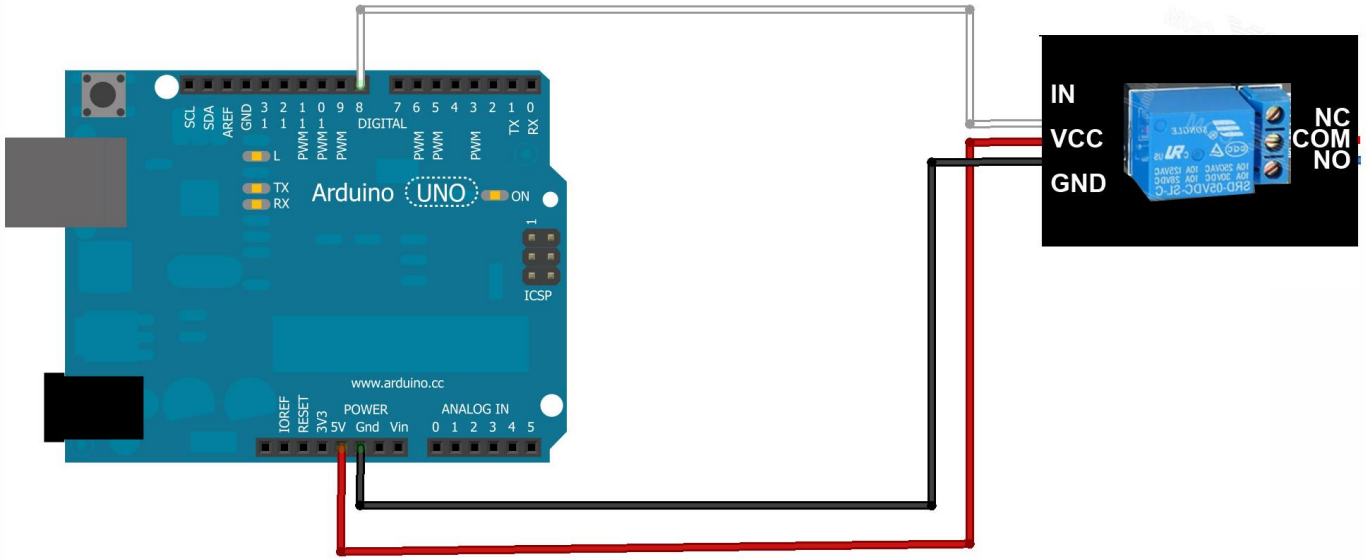
CIRCUIT CONTROL WITH RELAY

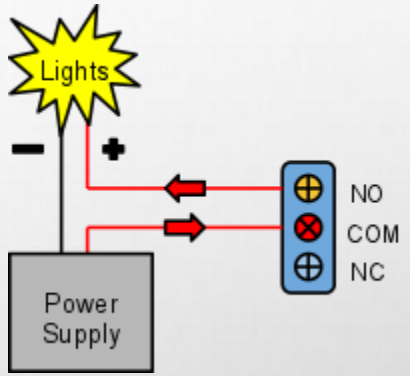


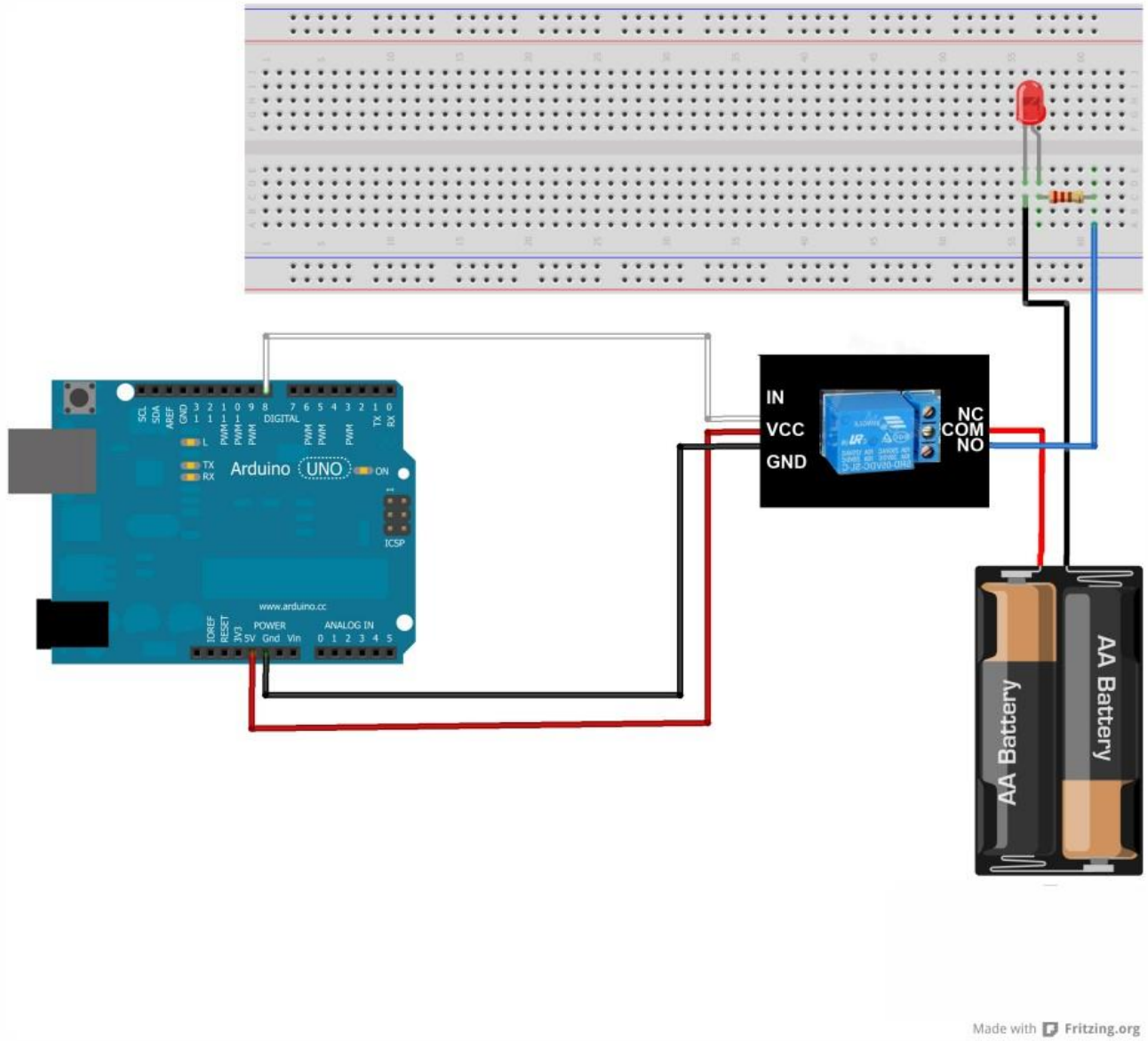


- https://www.youtube.com/watch?v=NYJHYKEOOGI

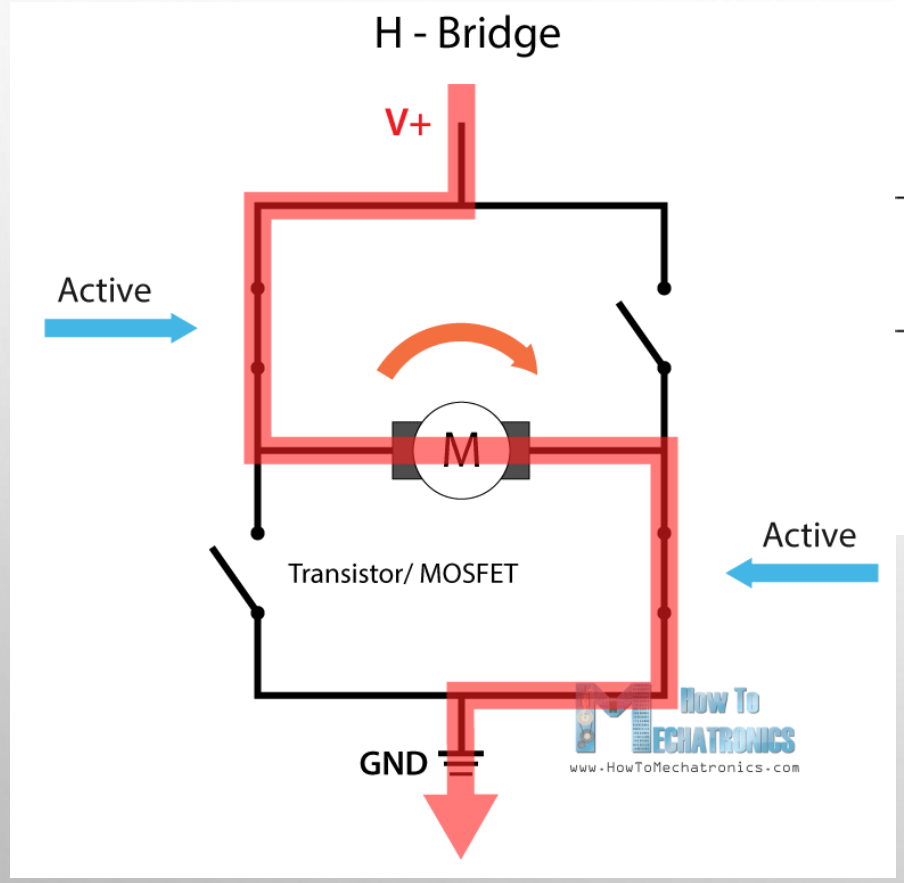
USING RELAYS

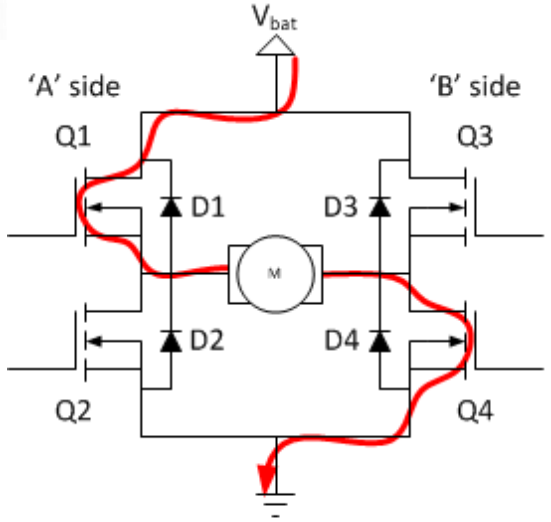


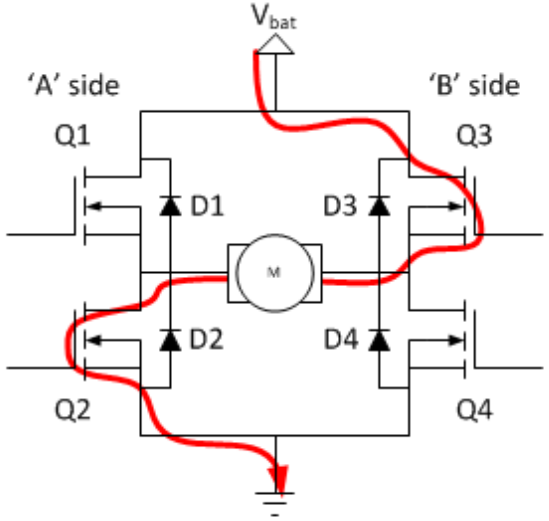



IN
VCC
GND
NC
COM
NO
Arduino
UNO
www.arduino.cc
AA Battery
AA Battery
Made with Fritzing.org

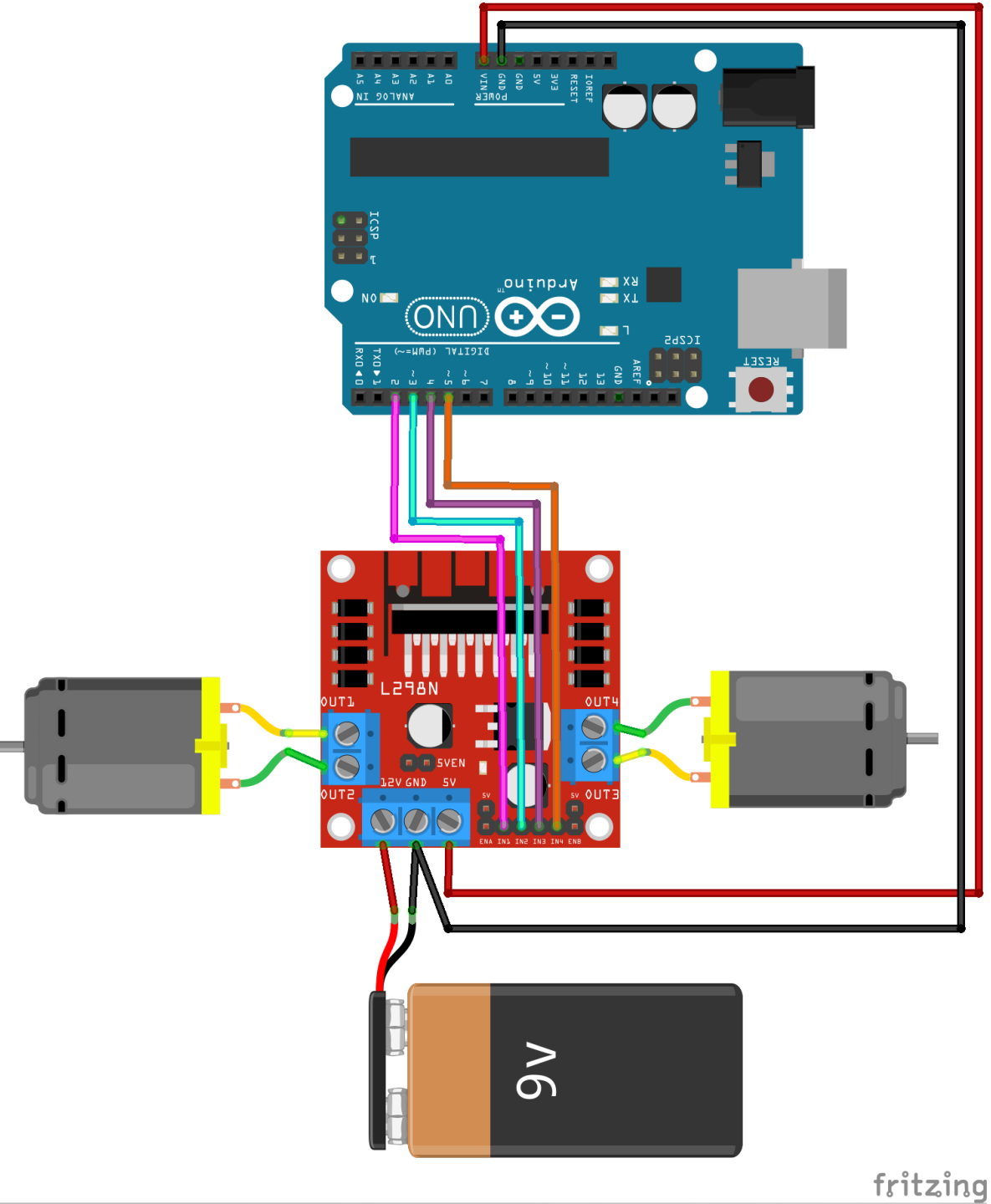
LOAD POLARITY SWITCH

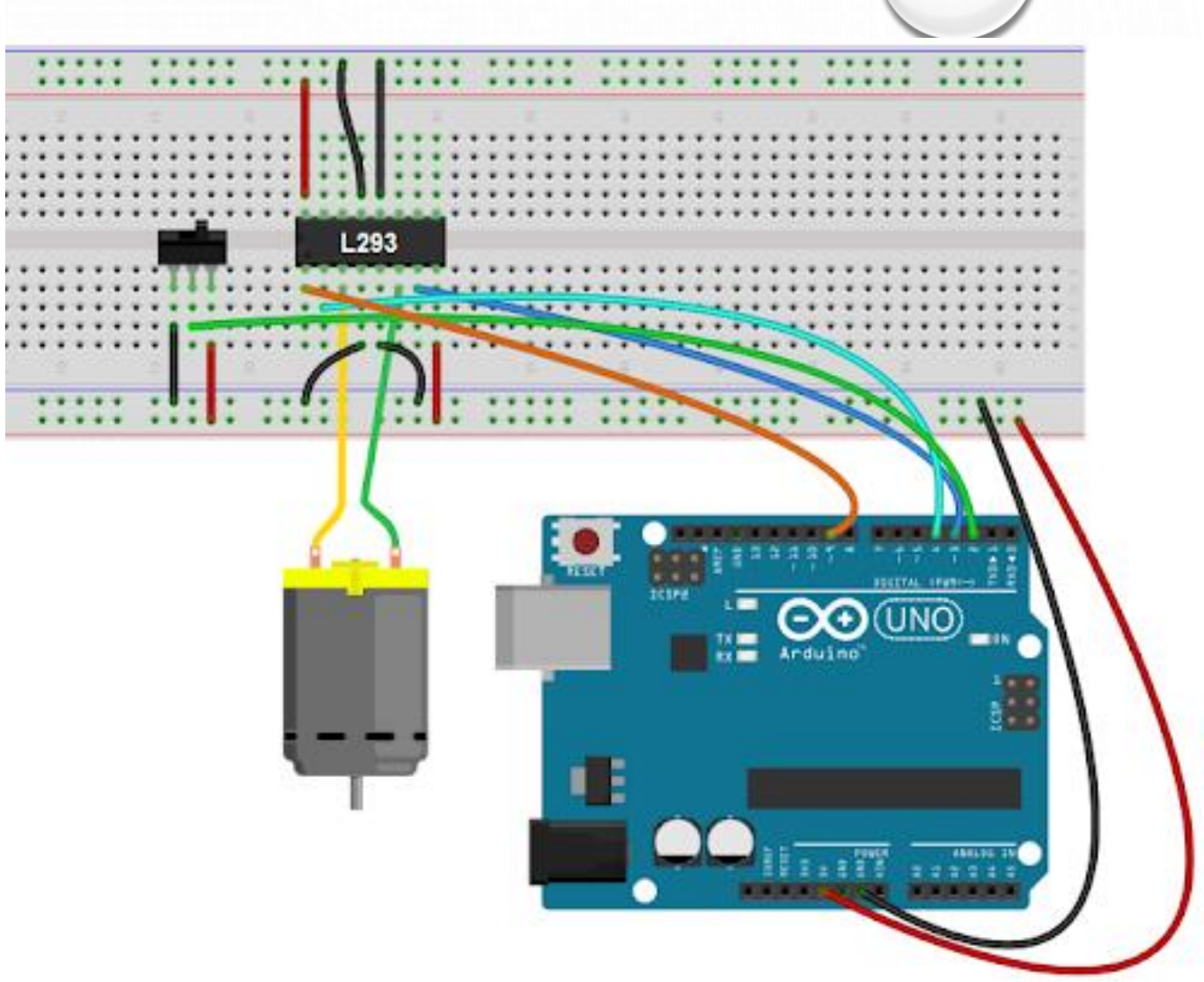






Q1	Q2	Q3	Q4	mode
close	open	open	open	motor coasts
close	open	open	close	forward motoring / move right
close	open	close	open	motor brakes (zero potential voltage)
open	close	open	open	motor coasts
open	close	open	close	motor brakes (zero potential voltage)
open	close	close	open	reverse motoring/ move left
open	open	open	open	motor coasts
open	open	open	close	motor coasts
open	open	close	open	motor coasts





HTTPS://WWW.YOUTUBE.COM/WATCH?V=I7IFSQ4TQU8

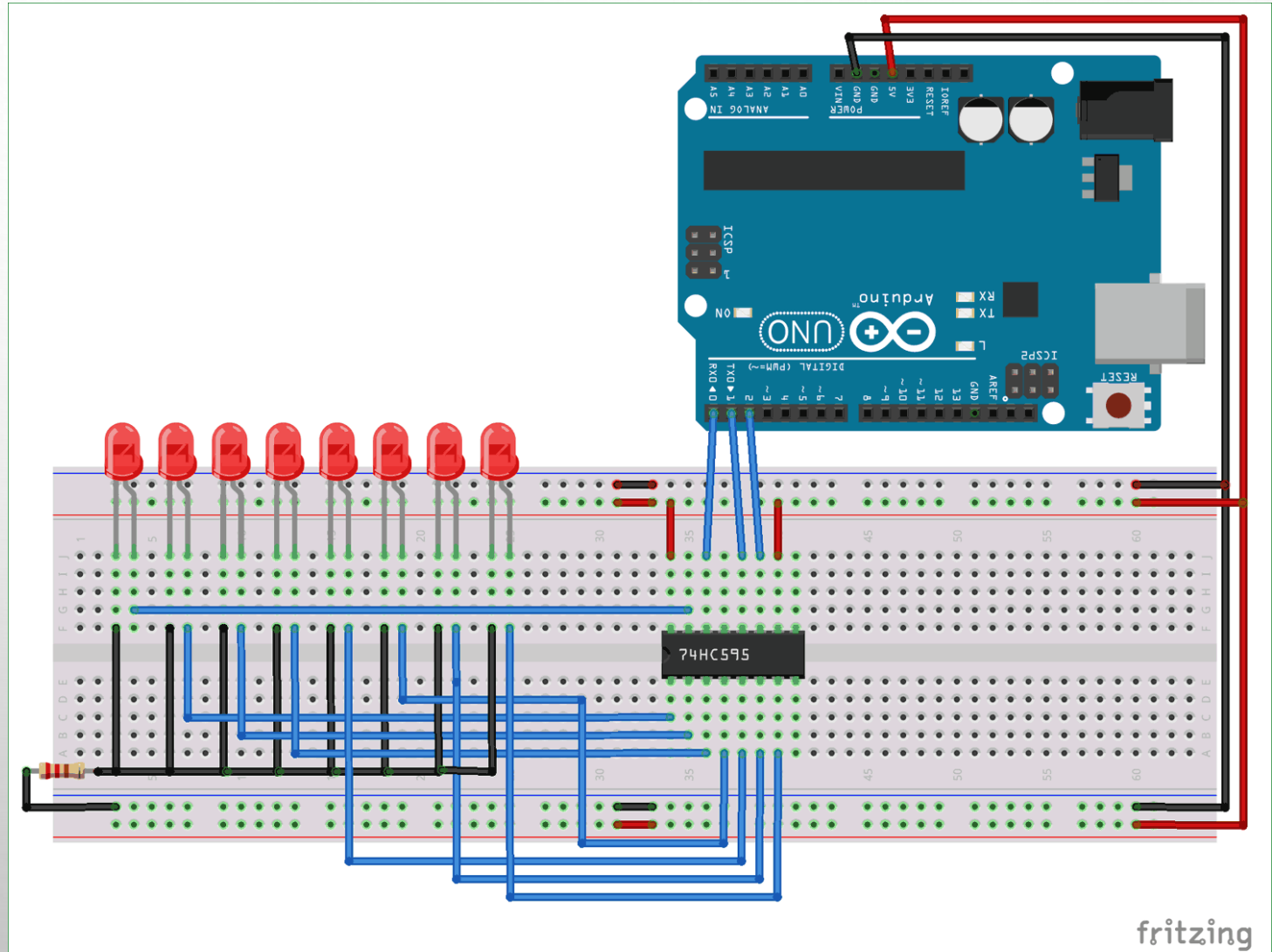
EXPANDING I/O CAPABILITIES

74HC595

8 BIT SERIAL IN

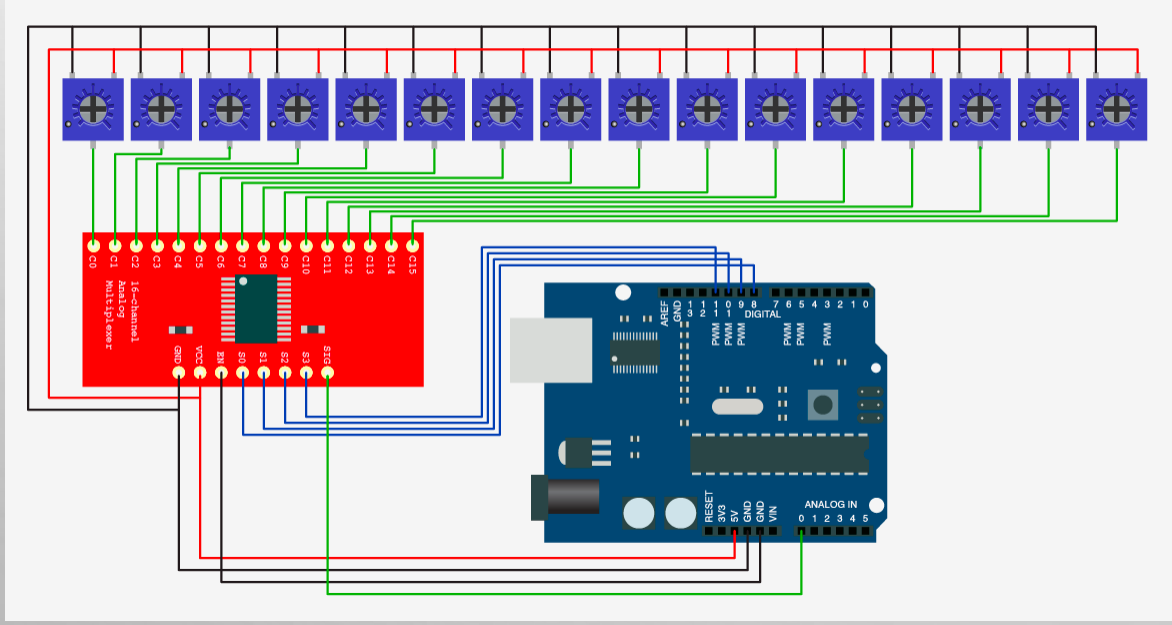
PARALLEL OUT

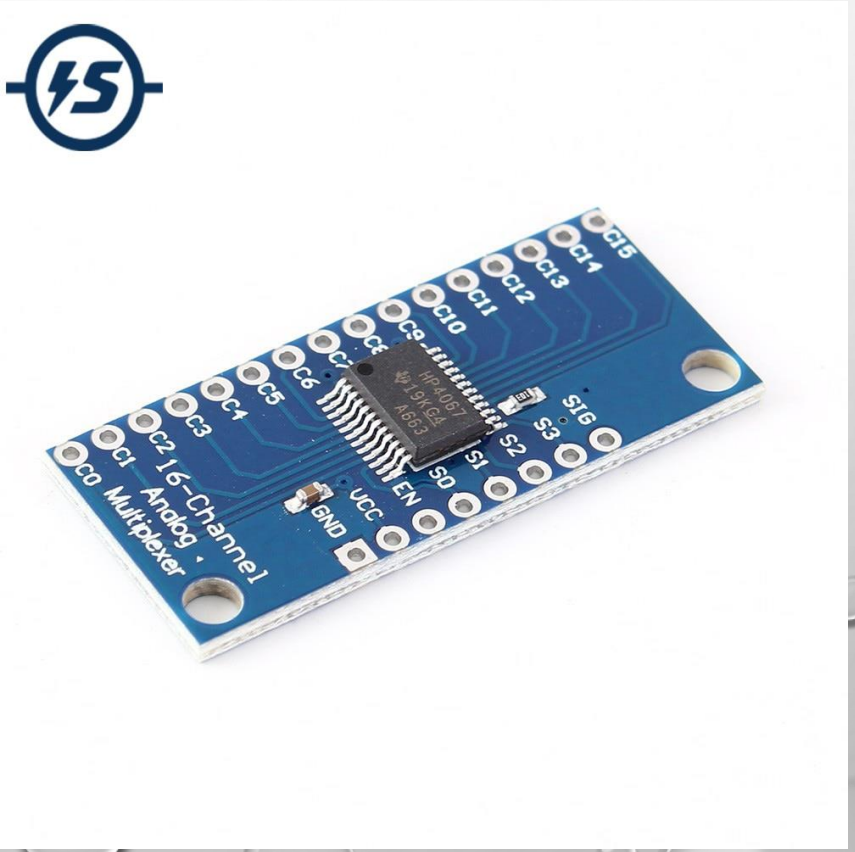
SHIFT REGISTER



MULTIPLEXER DEMULTIPLEXER

- MAIN DIFFERENCE FORM SHIFT REGISTERS ?

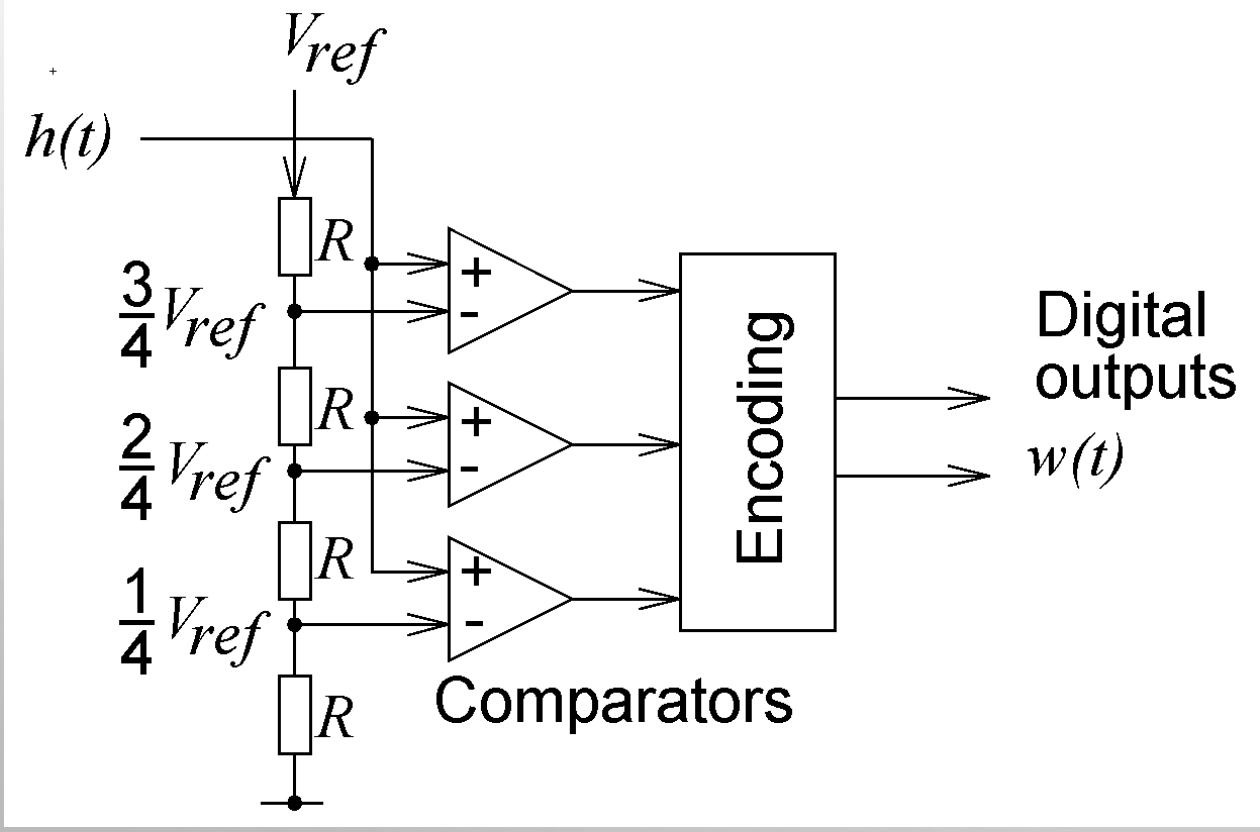


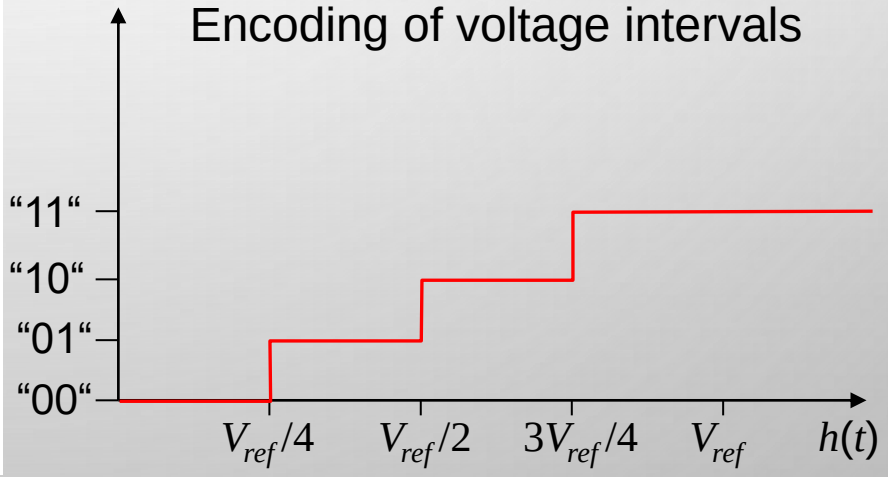


ADC / DAC

FLASH A/D CONVERTER

No decoding of $h(t) > V_{ref}$





RESOLUTION

- RESOLUTION (IN BITS): NUMBER OF BITS PRODUCED
- RESOLUTION Q (IN VOLTS): DIFFERENCE BETWEEN TWO INPUT VOLTAGES CAUSING THE OUTPUT TO BE INCREMENTED BY 1

$$Q = \frac{V_{FSR}}{n}$$ with

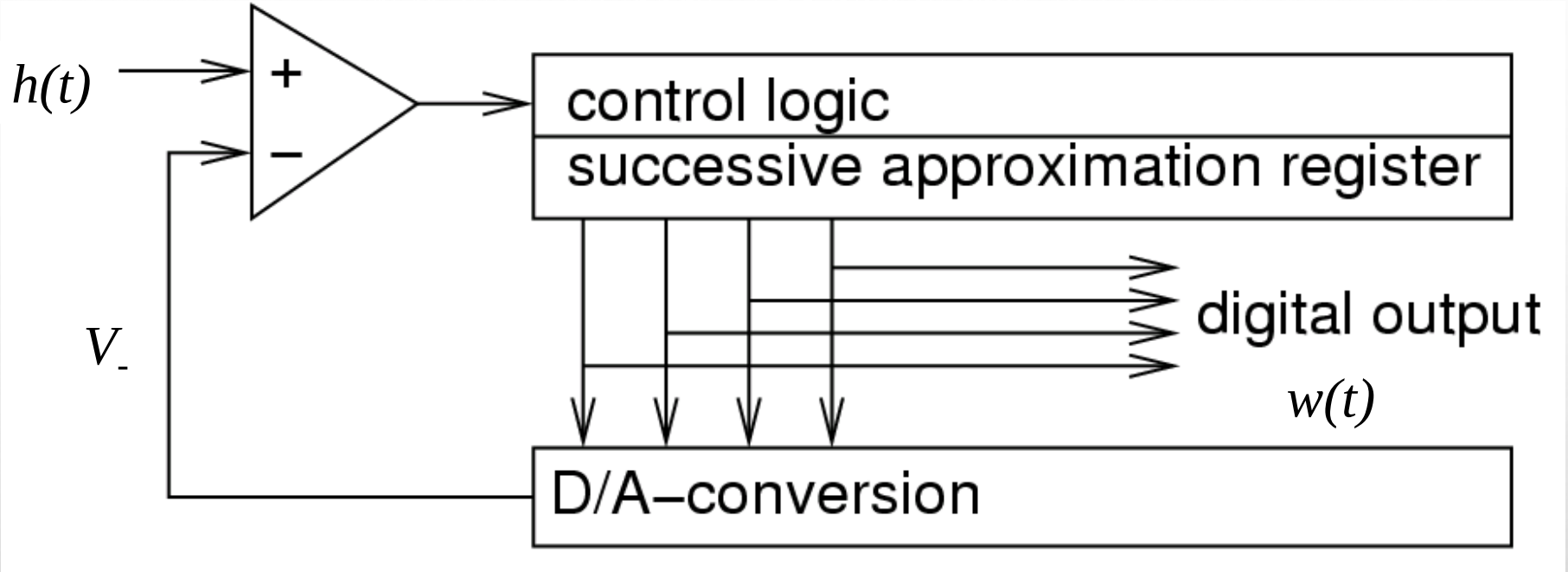
Q: resolution in volts per step
V_{FSR}: difference between largest and smallest voltage
n: number of voltage intervals

Example:
$Q = V_{ref}/4$ for the previous slide

RESOLUTION AND SPEED OF FLASH A/D-CONVERTER

- **PARALLEL COMPARISON WITH REFERENCE VOLTAGE**
- SPEED: $O(1)$
- HARDWARE COMPLEXITY: $O(N)$
- **APPLICATIONS**: E.G. IN VIDEO PROCESSING

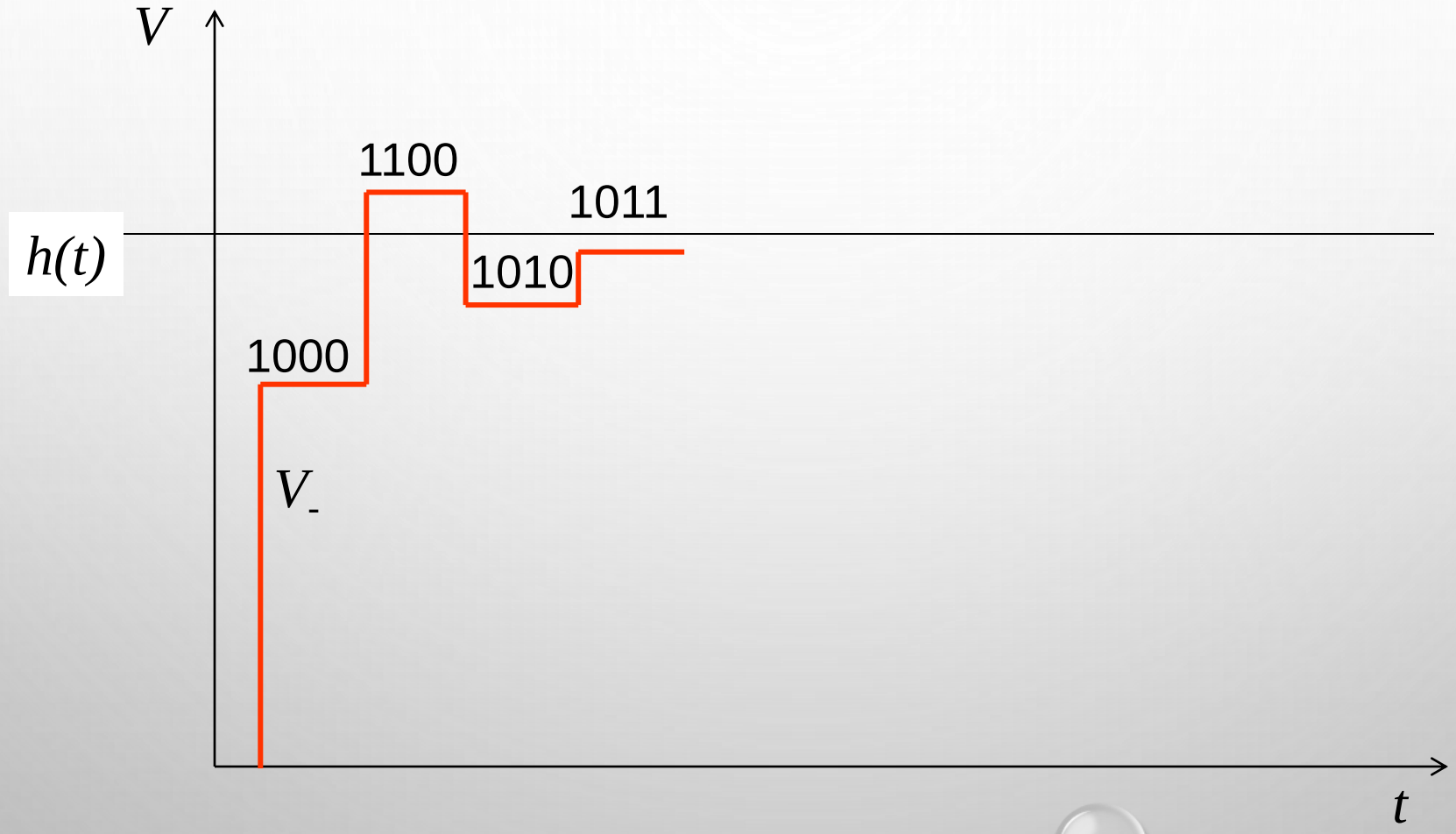
HIGHER RESOLUTION: SUCCESSIVE APPROXIMATION



- KEY IDEA: BINARY SEARCH:
- SET MSB='1'
- IF TOO LARGE: RESET MSB
- SET MSB-1='1'
- IF TOO LARGE: RESET MSB-1

Speed: $O(\log_2(n))$
Hardware complexity: $O(\log_2(n))$
with n= # of distinguished voltage levels;
slow, but high precision possible.

SUCCESSIVE APPROXIMATION (2)



PROGRAMMING MICROCONTROLLERS

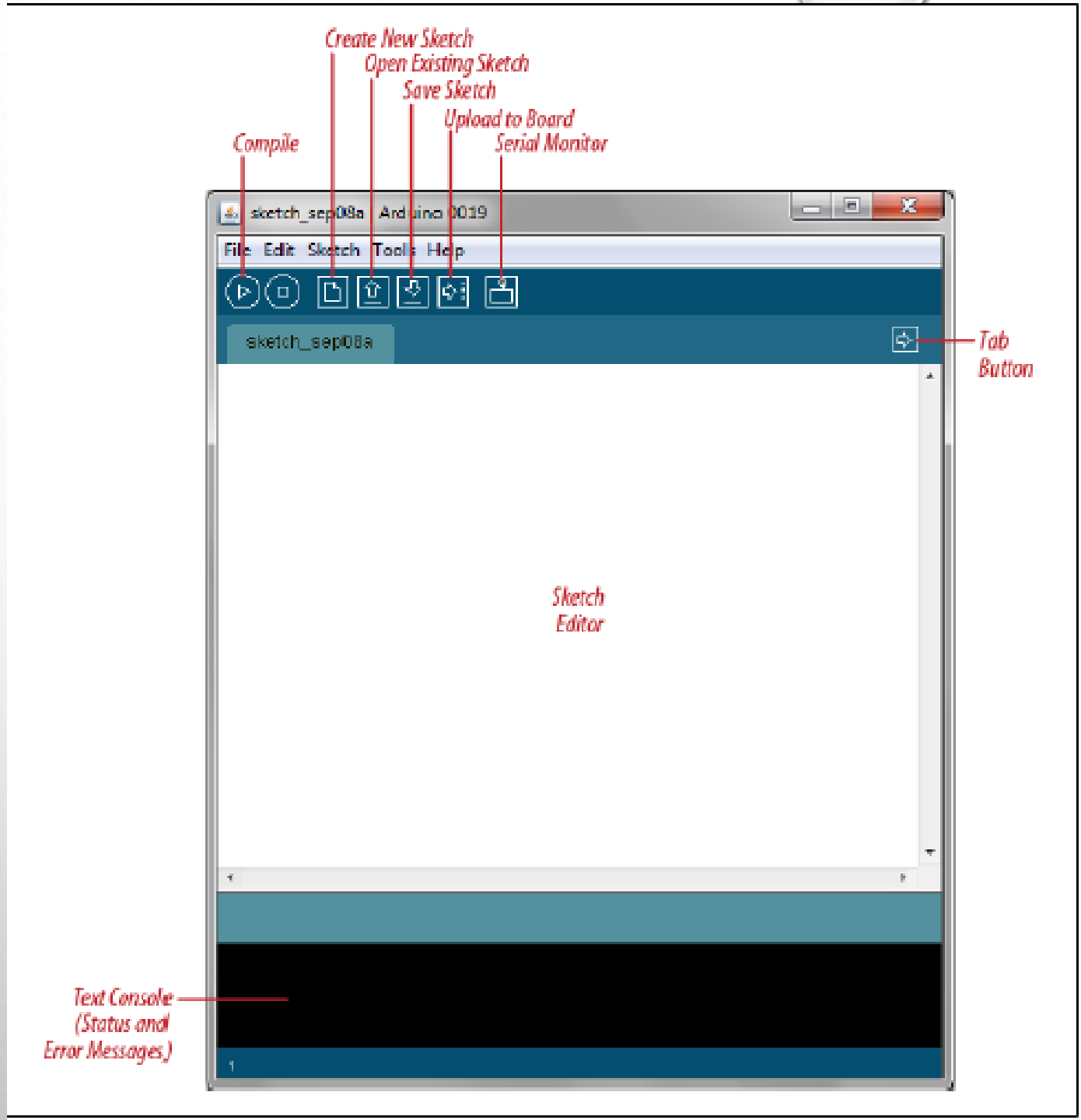
ARDUINO PLATFORM PROGRAMMING

AGENDA

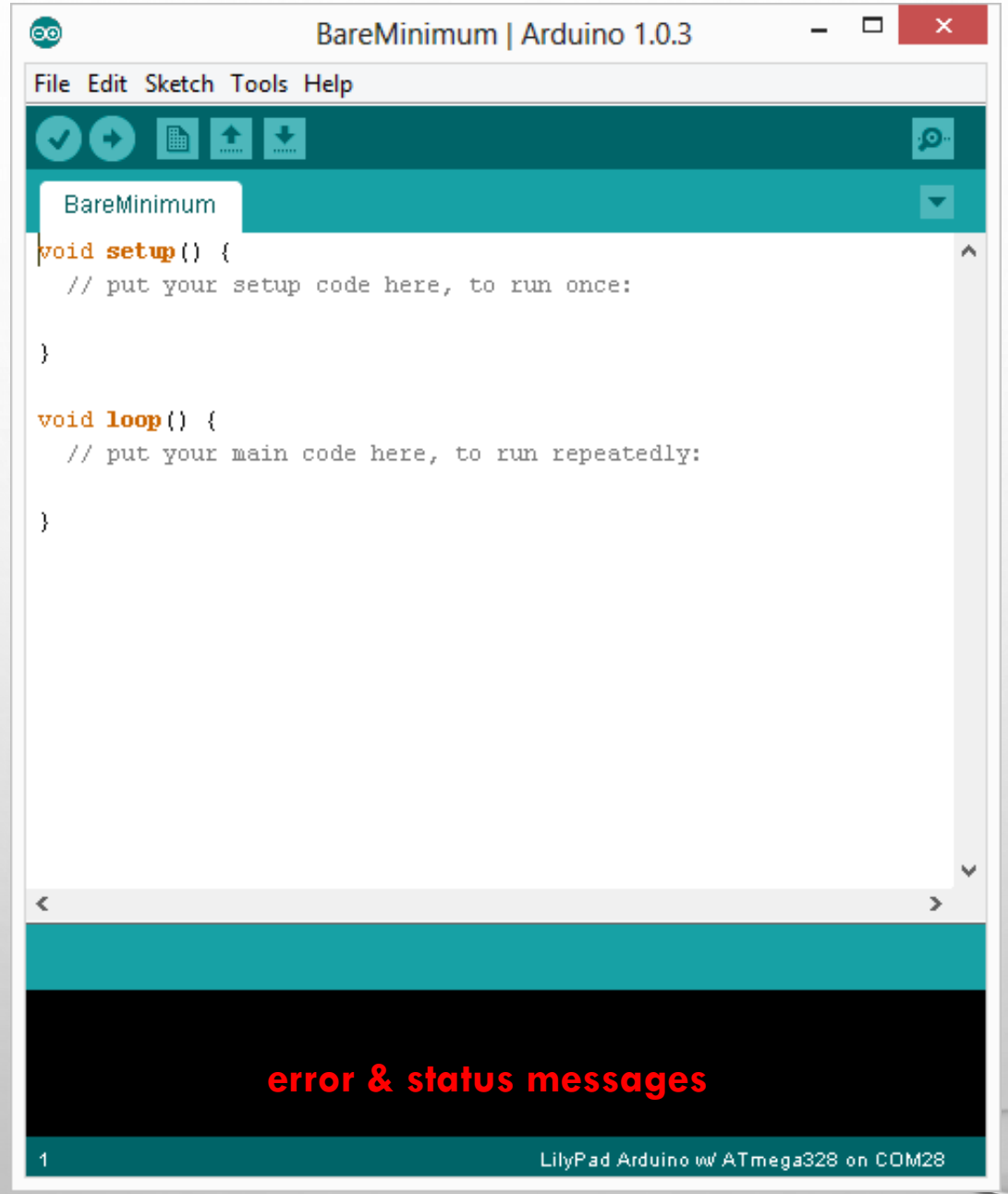
- ARDUINO IDE
- MAJOR CONCEPTS
- RANGE MAPPING
- INTERRUPTS
- DELAYS
- REMEMBERING CHANGES

ARDUINO IDE

- FILE HANDLER
- CODE EDITOR
- COMPILER
- CONSOLE



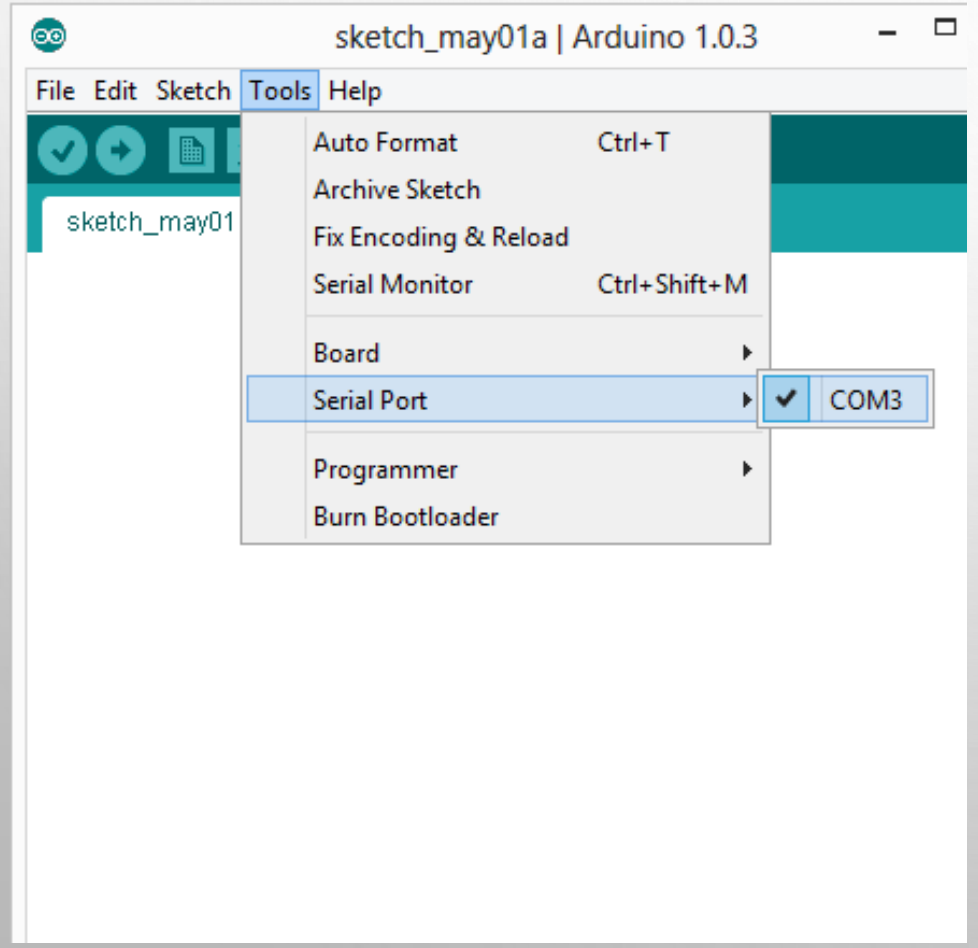
BARE MIN.



TWO REQUIRED FUNCTION METHODS / ROUTINES:

```
VOID SETUP()
{
      // RUNS ONCE
}

VOID LOOP()
{
      // REPEATS
}
```

SETTINGS: TOOLS → SERIAL PORT

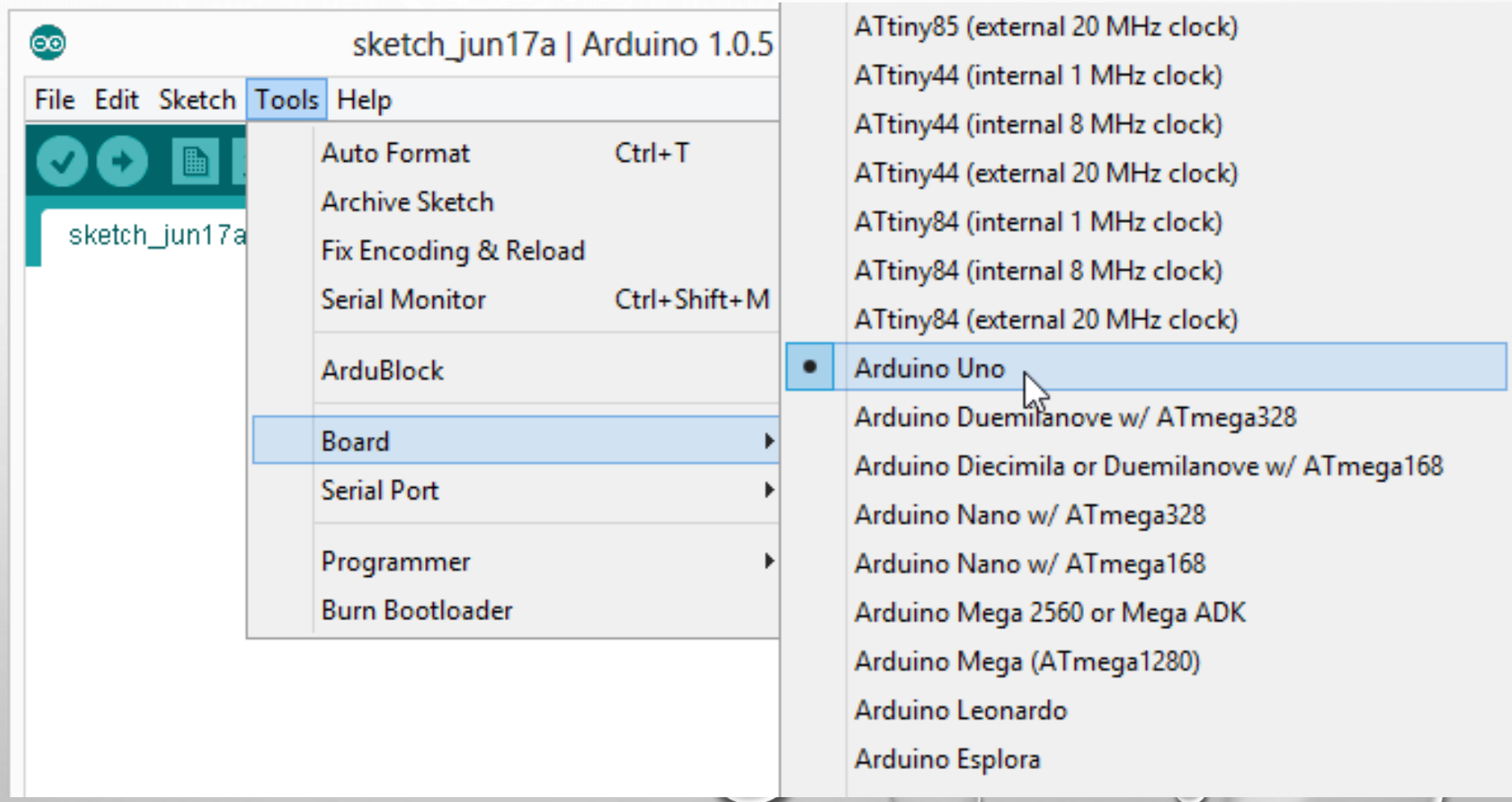


- YOUR COMPUTER COMMUNICATES TO THE ARDUINO MICROCONTROLLER VIA A SERIAL PORT → THROUGH A USB-SERIAL ADAPTER.

- CHECK TO MAKE SURE THAT THE DRIVERS ARE PROPERLY INSTALLED.

SETTINGS: TOOLS → BOARD

- NEXT, DOUBLE-CHECK THAT THE PROPER BOARD IS SELECTED UNDER THE TOOLS→BOARD MENU.



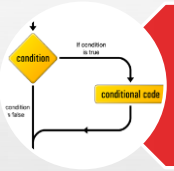
MAJOR CONCEPTS

- SERIAL.PRINTLN(VALUE);
  - PRINTS THE VALUE TO THE SERIAL MONITOR ON YOUR COMPUTER
- PINMODE(PIN, MODE);
  - CONFIGURES A DIGITAL PIN TO READ (INPUT) OR WRITE (OUTPUT) A DIGITAL VALUE

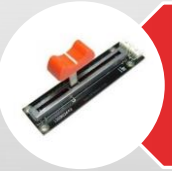
digitalWrite()

analogWrite()

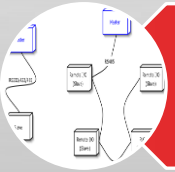
digitalRead()



if() statements / Boolean



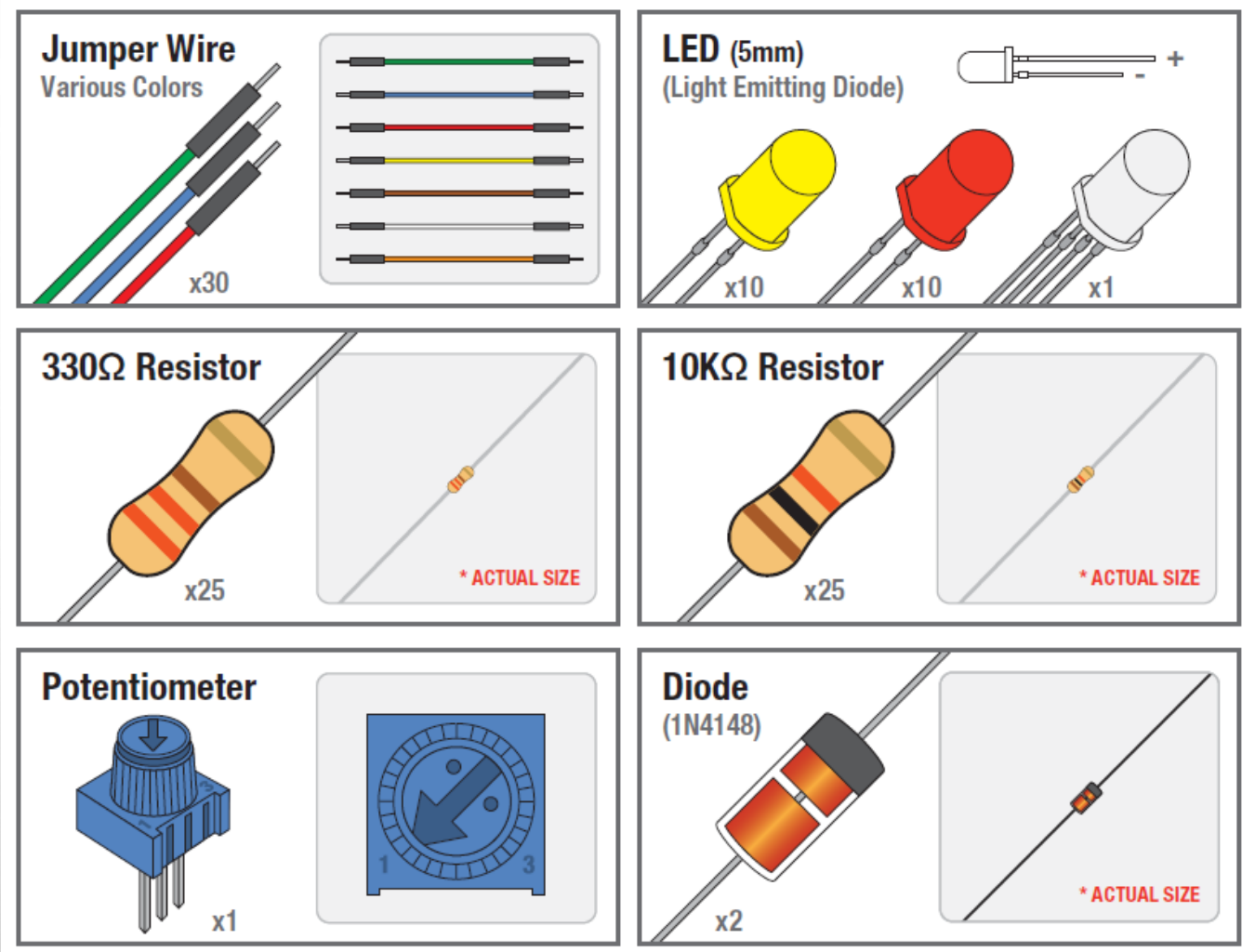
analogRead()



Serial communication

LEDS

- SIK COMPONENTS
- COMMON CATHODE RGB LED
- THESE 5MM UNITS HAVE FOUR PINS - CATHODE IS THE LONGEST PIN. ONE FOR EACH COLOR AND A COMMON CATHODE. USE THIS ONE LED FOR THREE STATUS INDICATORS OR PULSE WIDTH MODULATE ALL THREE AND GET MIXED COLORS!



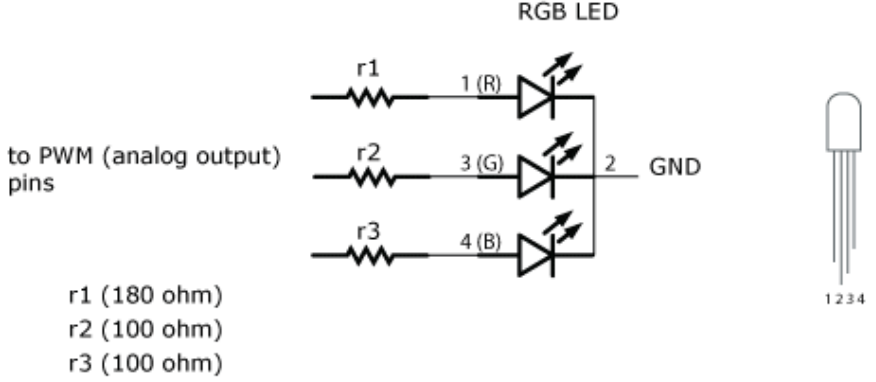
LEDS

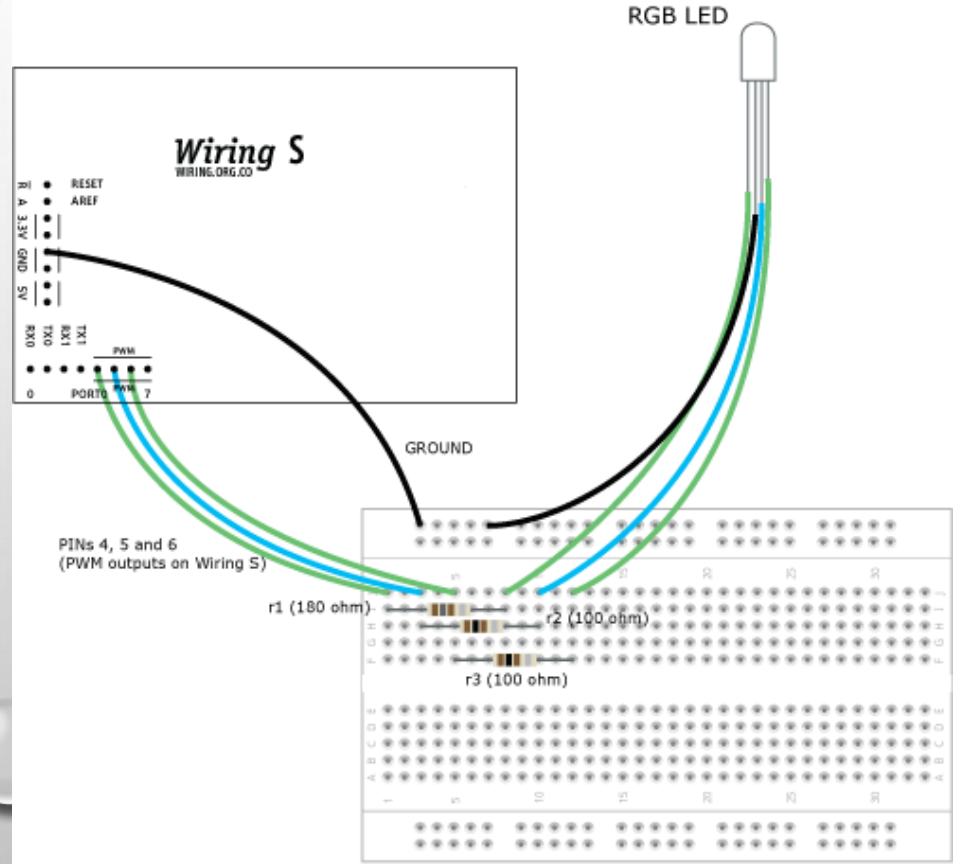
```
void setup()
{
pinMode(13, OUTPUT);   //configure pin 77 as output
blink1()
}
// blink an LED once
void blink1()
{
digitalWrite(13,HIGH); // turn the LED on
delay(500); // wait 500 milliseconds
digitalWrite(13,LOW); // turn the LED off
delay(500); // wait 500 milliseconds
}
```

```
VOID LOOP()  //BLINK A LED REPEATEDLY
{
DIGITALWRITE(13,HIGH); // TURN THE LED ON
DELAY(500); // WAIT 500 MILLISECONDS
DIGITALWRITE(13,LOW); // TURN THE LED OFF
DELAY(500); // WAIT 500 MILLISECONDS
}
```

RGB LED

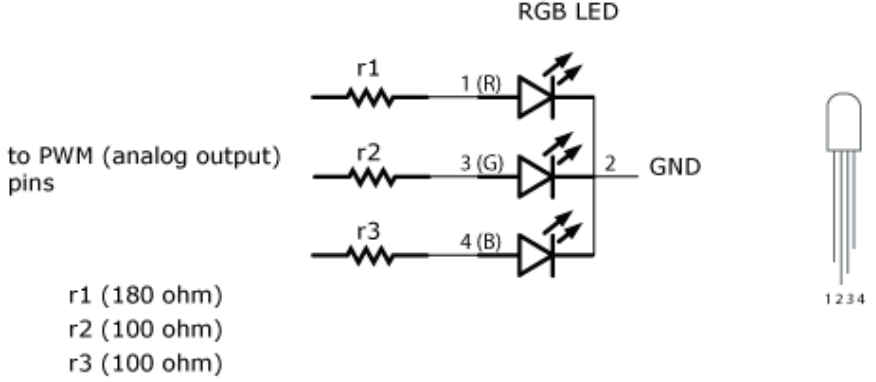
```
int REDPin = 4;     // RED pin of the LED to
PWM pin 4
int GREENPin = 5;   // GREEN pin of the LED
to PWM pin 5
int BLUEPin = 6;    // BLUE pin of the LED
to PWM pin 6
int brightness = 0; // LED brightness
int increment = 5;  // brightness increment
```

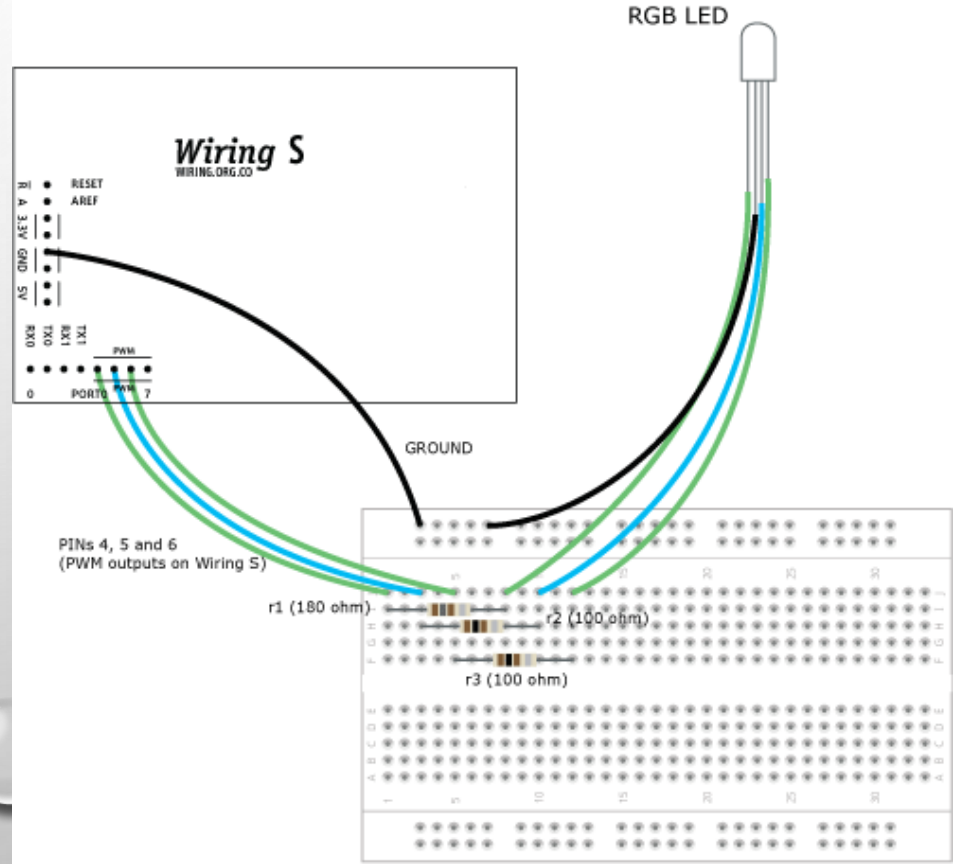




RGB LED

```
void setup()
{
  pinMode(REDPin, OUTPUT);
  pinMode(GREENPin, OUTPUT);
  pinMode(BLUEPin, OUTPUT);
  Serial.begin(9600);
}
```





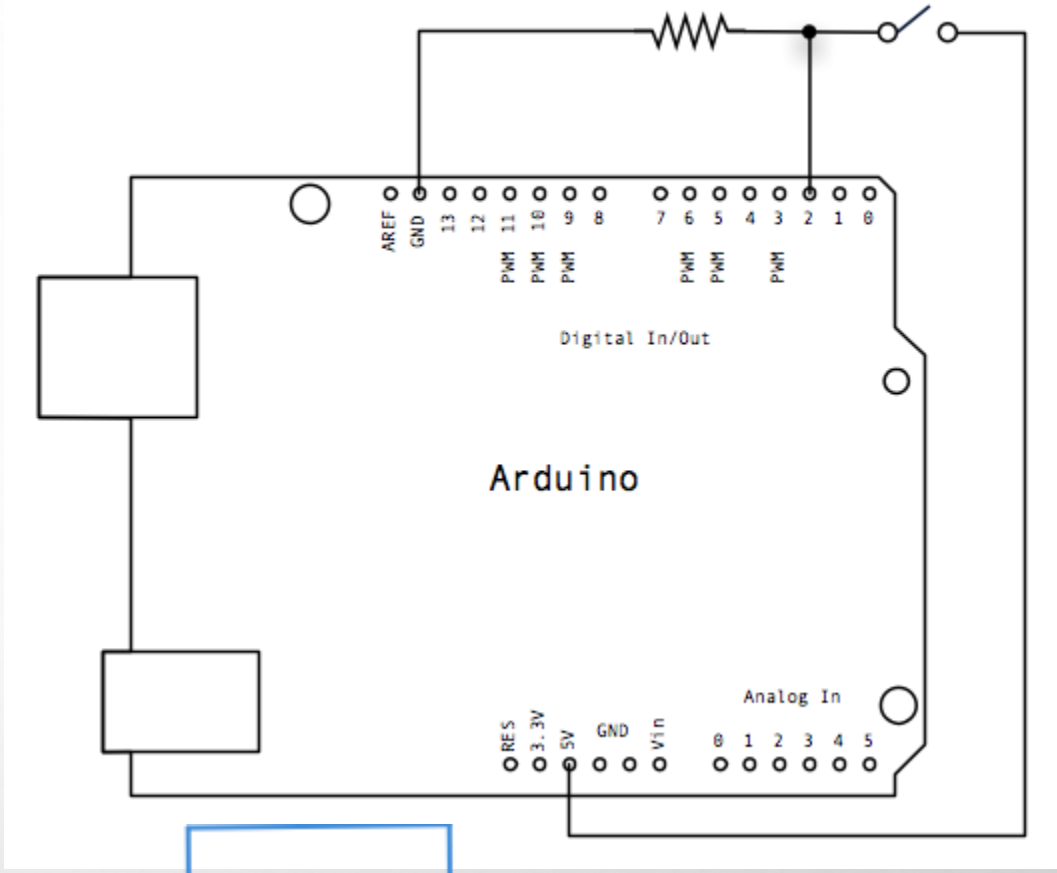
RGB LED

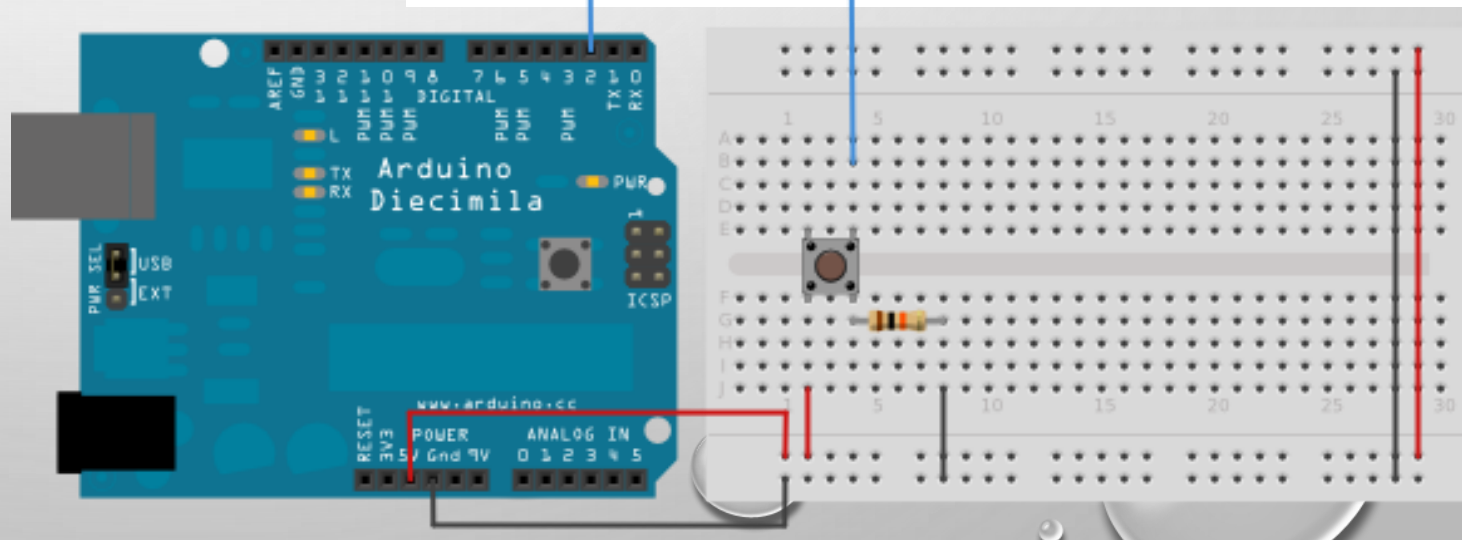
```
void loop()
{
  brightness = brightness + increment;  // increment brightness for next loop iteration

  if (brightness <= 0 || brightness >= 255)    // reverse the direction of the fading
  {
    increment = -increment;
  }
  brightness = constrain(brightness, 0, 255);
  analogWrite(REDPin, brightness);
  analogWrite(GREENPin, brightness);
  analogWrite(BLUEPin, brightness);

  delay(20);  // wait for 20 milliseconds to see the dimming effect
}
```

SWITCHES

```
CONST INT INPUTPIN = 2;
VOID SETUP() {
PINMODE(INPUTPIN, INPUT);
}
VOID LOOP(){
INT VAL = DIGITALREAD(INPUTPIN);
}
```

- WHAT DOES A RESISTOR DO ?





INTERRUPT PIN

INTERRUPT PINS OF UNO BOARD ARE 2,3

THEY ALLOW EXECUTION TO JUMP ONCE AN INPUT IS HIGH OR LOW (ACCORDING TO PULLUP/PULLDOWN MODES)

THE PROGRAM WILL CONTINUE EXECUTION FROM WHERE IT STOPPED

THE FOLLOWING CODE SWITCHES THE STATE BETWEEN LOW AND HIGH ONCE THE INTERRUPT PIN 2 HAS HIGH INPUT

```
const byte ledPin = 13;
const byte interruptPin = 2;
volatile byte state = LOW;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), blink, CHANGE);
}

void loop() {
  digitalWrite(ledPin, state);
}

void blink() {
  state = !state;
}
```

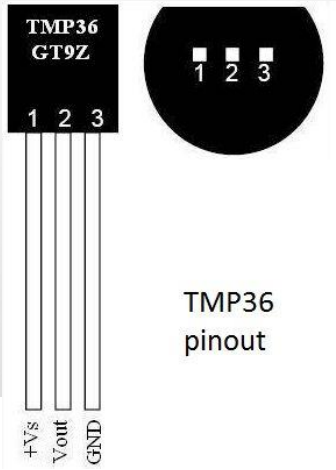
ANALOG INPUT – TEMPERATURE SENSOR

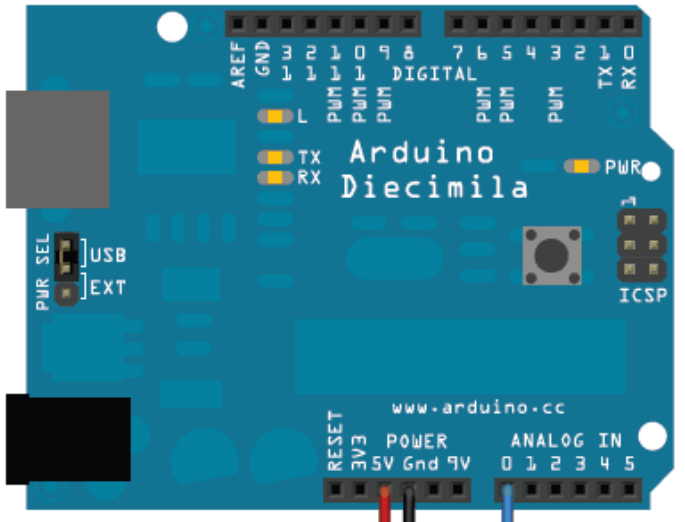
- TMP36
- **AD22100 TEMPERATURE SENSOR**
- OUTPUT V0 RANGING FROM 0.25 V AT −50°C TO +4.75 V AT +150°C
- ASSUME LINEAR RESPONSE AND TEMPERATURE COEFFICIENT OF 22.5 MV/°C , VOLTAGE AT 0°C IS 1.375
-

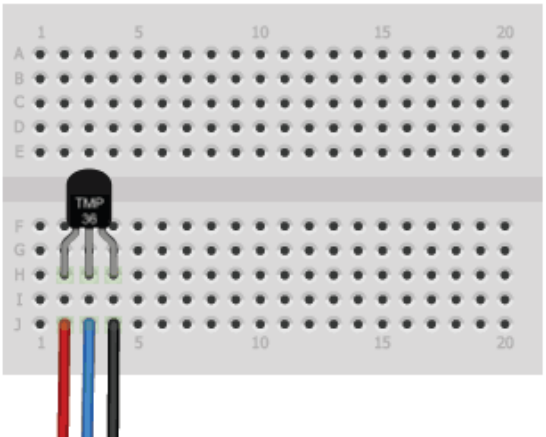
INT SENSORVALUE = ANALOGREAD(A0);

FLOAT VOLTAGE= SENSORVALUE * (5.0 / 1023);

TEMPERATURE = (FLOAT VOLTAGE-1.375) / 0.0225







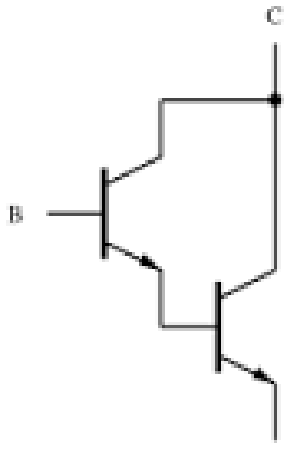
ANALOG INPUT – RANGE MAPPING

```
CONST INT POTPIN = 0; // SELECT THE INPUT PIN FOR THE POTENTIOMETER
VOID LOOP() {
INT VAL; // THE VALUE COMING FROM THE SENSOR
INT PERCENT; // THE MAPPED VALUE
VAL = ANALOGREAD(POTPIN); // READ THE VOLTAGE ON THE POT (VAL RANGES
  FROM 0 TO 1023)
PERCENT = MAP(VAL,0,1023,0,100); // PERCENT WILL RANGE FROM 0 TO 100.
```

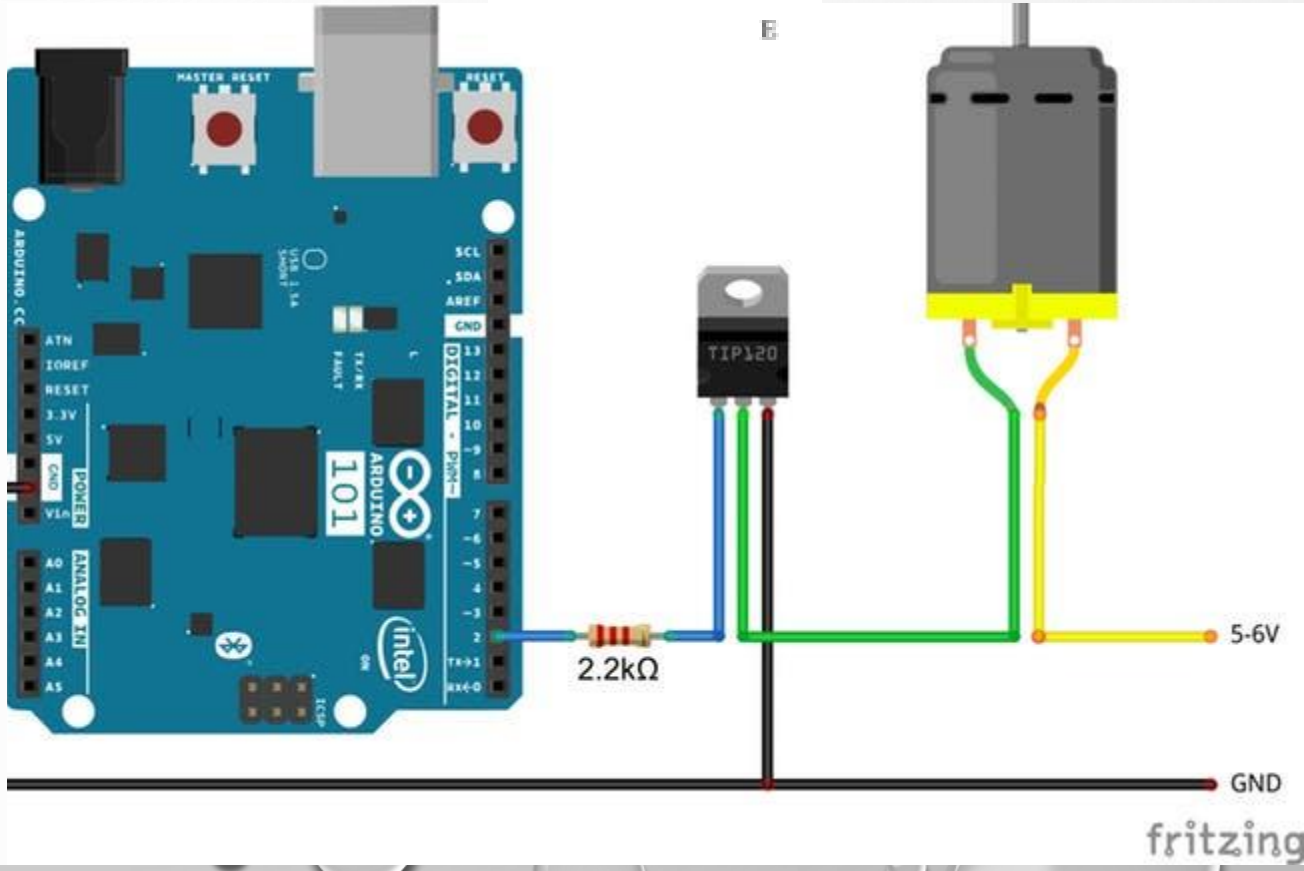
CONTROLLING DC MOTOR DIRECTION & SPEED

- OPEN DISCUSSION

CONTROLLING DC MOTOR

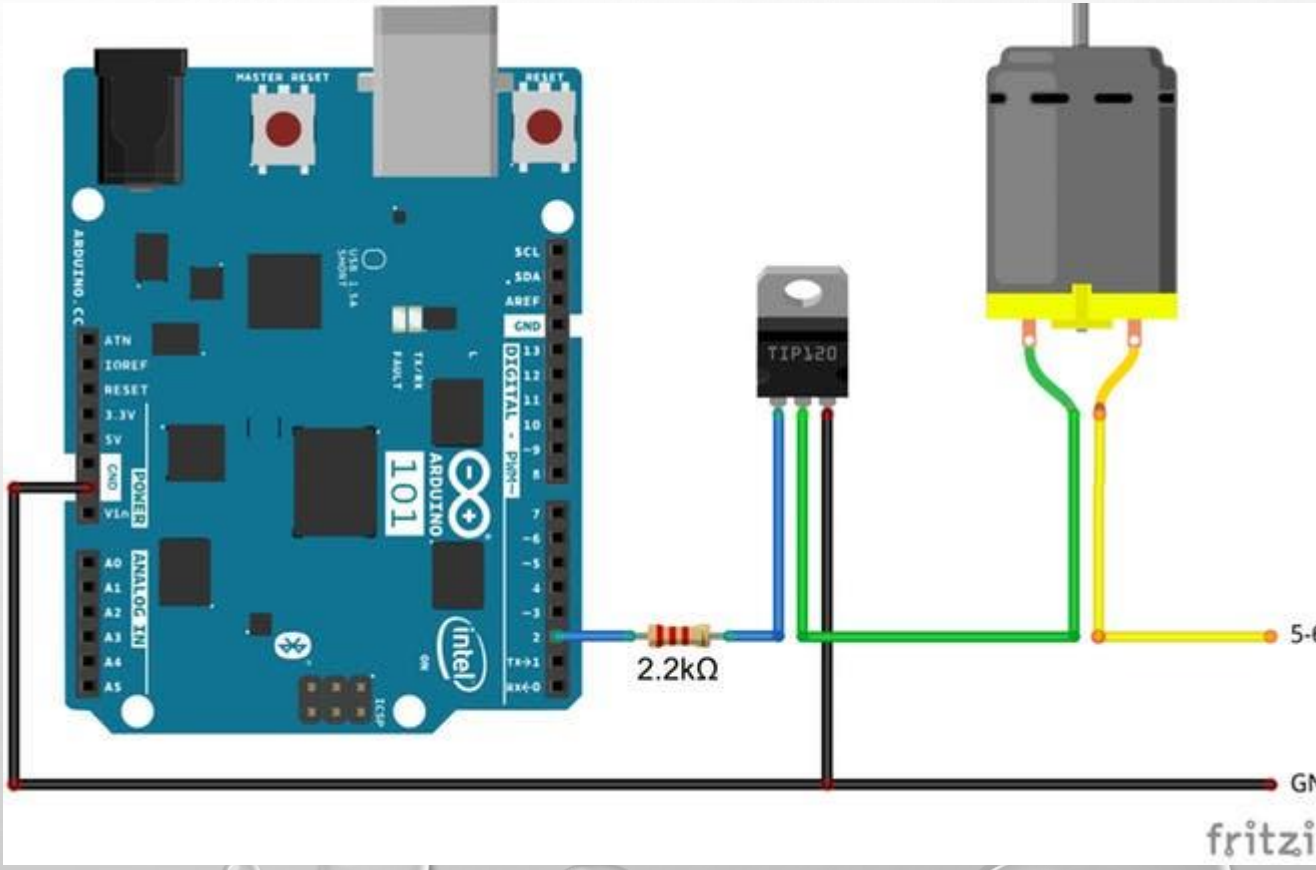



```
#define DC_MOTOR_PIN 2

void setup() {
  /* Initialize DC motor control pin as digital output */
  pinMode( DC_MOTOR_PIN, OUTPUT );
}

void loop() {
  /* Run motor */
  digitalWrite( DC_MOTOR_PIN, HIGH );
  /* Let it run for a while */
  delay( 2000 );
  /* Stop motor */
  digitalWrite( DC_MOTOR_PIN, LOW );
  /* Do nothing */
  while( 1 );
}
```



CONTROLLING DC MOTOR SPEED

```
#define DC_MOTOR_PIN 3

void setup() {
  /* Initialize DC motor control pin as digital output */
  pinMode( DC_MOTOR_PIN, OUTPUT );
}

void loop() {
  /* Gradually increase motor speed to full speed */
  for( int i = 0; i < 255; i = i+5 )
    analogWrite( DC_MOTOR_PIN, i );
  /* Gradually decrease motor speed to half speed */
  for( int i = 255; i > 128; i = i-5 )
    analogWrite( DC_MOTOR_PIN, i );
  /* Stop the motor */
  analogWrite( DC_MOTOR_PIN, 0 );
  /* Do nothing */
  while( 1 );
}
```



THE BUILD PROCESS

- **CODE VERIFICATION C / C++ SYNTAX CHECK**
- **TRANSFORMATIONS (INCLUDING ARDUINO.H**
- **CODE COMPILATION AVR-GCC -> MACHINE BASED INSTRUCTIONS .O FILE**
- **LINKAGE WITH LIBRARIES (ORIGINAL IMPLEMENTATIONS) .HEX FILE**
- **UPLOAD INTO BOARD FLASH MEMORY - AVRDUDE**

DELAY

- DELAY

BASIC DELAY

```
void setup() {
  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
}
```

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


void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  for (int x=0; x < 1000; x++) {     // Wait for 1 second
    delay(1);
  }
  digitalWrite(13, LOW);   // set the LED on
  for (int x=0; x < 1000; x++) {     // Wait for 1 second
    delay(1);
  }
}
```

```
void setup() {
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
}
 void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  for (int x=0; x < 1000; x++) {          // wait for a secoond
    delay(1);
    if (x==500) {
      digitalWrite(12, HIGH);
    }
  }
  digitalWrite(13, LOW);    // set the LED off
  for (int x=0; x < 1000; x++) {          // wait for a secoond
    delay(1);
    if (x==500) {
        digitalWrite(12, LOW);
    }
  }
}
```

```ino
unsigned long interval=1000; // the time we need to wait
unsigned long previousMillis=0; // millis() returns an unsigned long.

bool ledState = false; // state variable for the LED

void setup() {
 pinMode(13, OUTPUT);
 digitalWrite(13, ledState);
}

void loop() {
 unsigned long currentMillis = millis(); // grab current time

 // check if "interval" time has passed (1000 milliseconds)
 if ((unsigned long)(currentMillis - previousMillis) >= interval) {

   ledState = !ledState; // "toggles" the state
   digitalWrite(13, ledState); // sets the LED based on ledState
   // save the "current" time
   previousMillis = millis();
 }
}
```

REMEMBERING CHANGES

- THE SUPPORTED MICRO-CONTROLLERS ON THE UNO ARDUINO BOARDS HAVE DIFFERENT AMOUNTS OF EEPROM: 1024 BYTES ON THE ATMEGA328
- EPROM IS MEMORY WHOSE VALUES ARE KEPT WHEN THE BOARD IS TURNED OFF (LIKE A TINY HARD DRIVE).
- YOU CAN READ , WRITE , CLEAR , UPDATE
- EEPROM UPDATE: STORES VALUES READ FROM A0 INTO EEPROM, WRITING THE VALUE ONLY IF DIFFERENT, TO INCREASE EEPROM LIFE.

REMEMBERING CHANGES

EEPROM

write()

Description

Write a byte to the EEPROM.

Syntax

EEPROM.write(address, value)

Parameters

address: the location to write to, starting from 0 (*int*)

value: the value to write, from 0 to 255 (*byte*)

```
#include <EEPROM.h>
int addr = 0;
void setup() {}
void loop() {
int val = analogRead(0) / 4;
 EPROM.write(addr, val);
  addr = addr + 1;
  if (addr == EEPROM.length()) {
    addr = 0;
  }
  delay(100);
}
```

REFERENCES

- HTTPS://WWW.BALDENGINEER.COM/MILLIS-TUTORIAL.HTML
- HTTPS://WWW.ARDUINO.CC/EN/REFERENCE/MILLIS
- HTTPS://THEALARMCLOCKSIXAM.COM/2012/01/22/OBJECT-ORIENTED-BLINKING-WITH-ARDUINO/