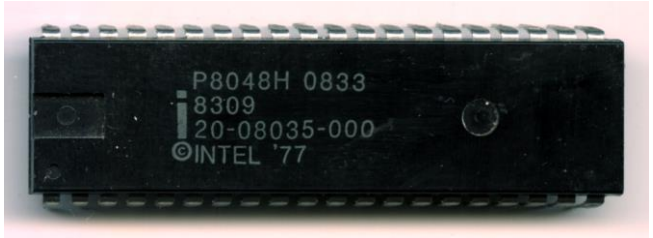
ECE 442/510
Internet of Things and Cyber Physical Systems

Lecture 3: Design with Arduino & Lab 1
Summer 2022

Jafar Saniie & Won-Jae Yi

What is a Microcontroller?



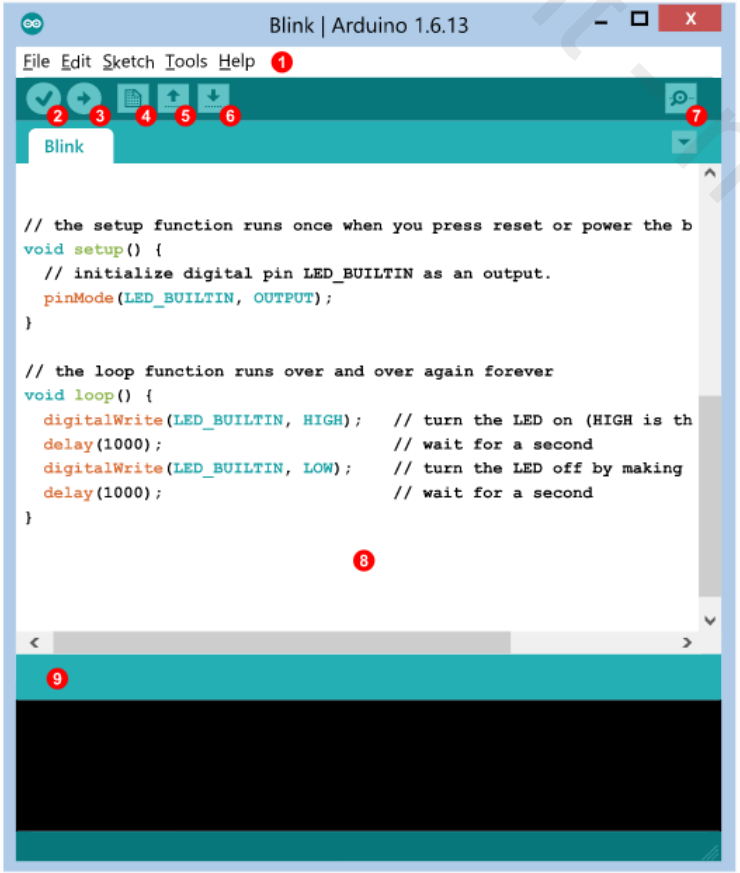
- Computer on a single integrated chip
  - CPU, Control Unit, Memory, Clock, I/O (see previous lecture notes)
- Common microcontrollers (to name a few…)
  - Atmel AT89, ATmega, AVR32, MARC4…
  - Freescale Semiconductor 68HCxx, 683xx…
  - Intel 8048, 8051, MCS-251, MCS-296…
  - Lattice Semiconductor Mico8, Mico32…
  - MSP430 - Microchip PIC10, PIC16, PIC24, PIC32…
  - ARM (from multiple manufacturers)
- Home appliances, Mobile devices, Automotive, Military, Gaming Consoles...

What is an Arduino?

- A physical piece of hardware

What is an Arduino?

- A programming environment



1. **Menu:** Selections of software features.
2. **Verify:** Compiles and verifies your sketch.
3. **Upload:** Send your sketch to STEMTera™ Breadboard.
4. **New:** Opens a new sketch window.
5. **Open:** Open and existing sketch.
6. **Save:** Save current active sketch.
7. **Monitor:** Opens a window to send and receive information.
8. **Editor:** Code editor area. Type your sketch in this area.
9. **Message:** IDE reports success or failure messages here.

What is an Arduino?

- A community & philosophy

Short History

- Initial team: Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, David Mellis
- Influenced by *Processing* Development Environment and *Wiring* Programming Language
  - Simple, low cost tools for creating digital projects by non-engineers
- *Wiring* platform had ATmega168, an IDE based on *Processing* and library functions
- Arduino project forked from *Wiring* platform by adding support to ATmega8
- Adafruit Industries supplies Arduino boards and parts

Various Types of Arduino

- Entry Level

Various Types of Arduino

- Arduino UNO
  - For general purpose development with electronics and coding
  - ATmega328, 5V, 14 Digital I/Os with 6 PWM channels
- Arduino Leonardo
  - Ideal for projects requiring the board to behave as a USB HID (human interface device)
  - ATmega32u4, 5V, 20 Digital I/Os with 7 PWM channels
- Arduino 101 (Intel Curie)
  - Intel x86 (Quark) & 32-bit ARC, 3.3V, 14 digital I/Os with 6 PWM channels
  - Bluetooth LE, 6-axis accelerometer/gyro
- Arduino Esplora
  - Arduino Leonardo based board with integrated sensors and actuators
  - ATmega32u4, 5V, analog joystick and 4 push buttons
- Arduino Micro, Nano, Mini
  - Smaller form factor, consumes reasonably less power

Various Types of Arduino

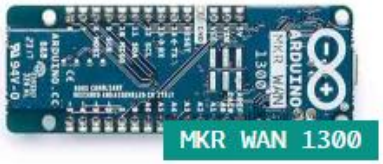
- Enhanced Features

Various Types of Arduino

- Arduino Mega 2560
  - Designed for more complex projects (3D printers and robotics)
  - ATmega2560, 54 digital I/Os with 15 PWM, 16 analog input pins
- Arduino Zero, M0/M0 Pro
  - Smart IoT devices, wearable technology, high-tech automation, etc.
  - 32-bit ARM Cortex M0, 20 digital I/Os, 6 12-bit ADC channels, 1 10-bit DAC channel
- Arduino Due
  - First Arduino board based on a 32-bit ARM core microcontroller
  - 32-bit ARM Cortex-M3, 54 digital I/Os, 12 analog inputs, 2 DAC
- Arduino MKR Zero
  - I^2S (Inter-IC Sound) bus, SD for sound, music & digital audio data

Various Types of Arduino

- Internet of Things



Various Types of Arduino

- Arduino Yun/Ethernet
  - Specifically designed for IoT development
- Arduino Tian
  - Integrated TCP/IP Ethernet Controller
  - ATmega328, 14 digital I/Os, 6 analog inputs, microSD card reader
- Arduino Industrial 101
  - Small form-factor YUN designed for product integration
  - ATmeag32u4, supports OpenWRT, built-in Wi-Fi,
- Arduino MKR 1000, FOX 1200, WAN 1300, GSM 1400
  - Ideal solutions for designing IoT devices with networking
  - 32-bit ARM Cortex-M0+
- Arduino Fio
  - Small and Li-Po battery ready for IoT devices
  - XBee-ready development board with ATmega328P

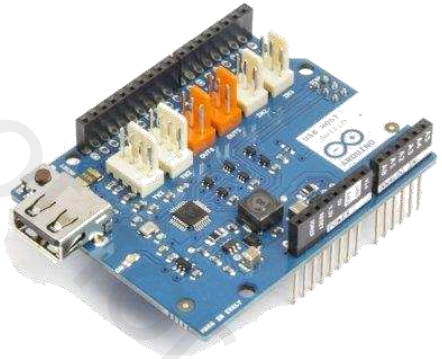
Various Types of Arduino

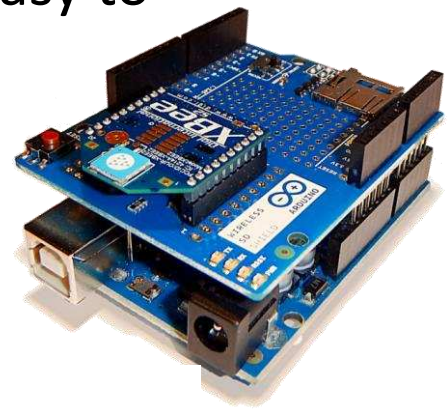
ENTRY LEVEL	UNO, LEONARDO, 101, ESPLORA, MICRO, NANO, MINI, MKR2UNO ADAPTER, STARTER KIT, LCD SCREEN
ENHANCED FEATURES	MEGA, ZERO, DUE, MEGA ADK, M0, M0 PRO, MKR ZERO, MOTOR SHIELD, USB HOST SHIELD, PROTO SHIELD, MKR PROTO SHIELD, 4 RELAYS SHIELD, MEGA PROTO SHIELD, MKR RELAY PROTO SHIELD, ISP, USB2SERIAL MICRO, USB2SERIAL CONVERTER
INTERNET OF THINGS	YÚN, ETHERNET, TIAN, INDUSTRIAL 101, LEONARDO ETH, MKR FOX 1200, MKR WAN 1300, MKR GSM 1400, MKR1000, YUN MINI, YÚN SHIELD, WIRELESS SD SHIELD, WIRELESS PROTO SHIELD, ETHERNET SHIELD V2, GSM SHIELD V2, MKR IoT BUNDLE
EDUCATION	CTC 101
WEARABLE	GEMMA, LILYPAD ARDUINO USB, LILYPAD ARDUINO MAIN BOARD, LILYPAD ARDUINO SIMPLE, LILYPAD ARDUINO SIMPLE SNAP
3D PRINTING	MATERIA 101

BOARDS | MODULES | SHIELDS | KITS | ACCESSORIES | COMING NEXT

Arduino Shields

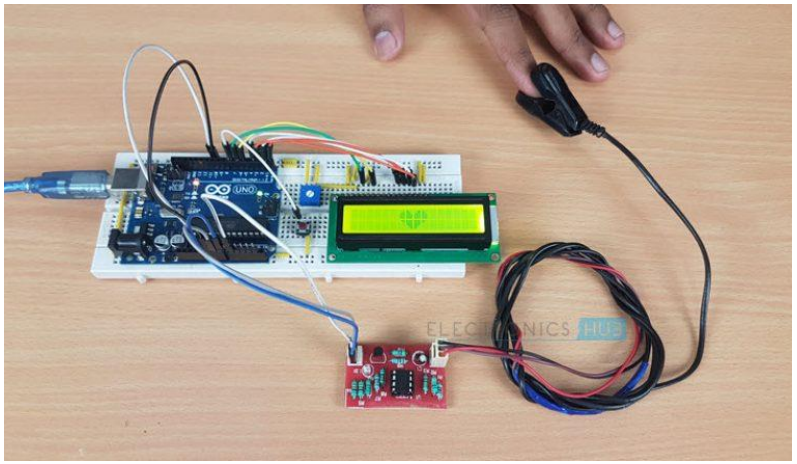
- Boards that can be plugged on top of Arduino PCB extending its capabilities
- Different shields follow the same philosophy; easy to mount, cheap to produce
- TFT Touchscreen Shield
- Motor/Servo Shield
- Ethernet Shield
- Audio Wave Shield
- Cellular/GSM Shield
- Bluetooth/Wi-Fi Shield
- Proto-shield and many more…

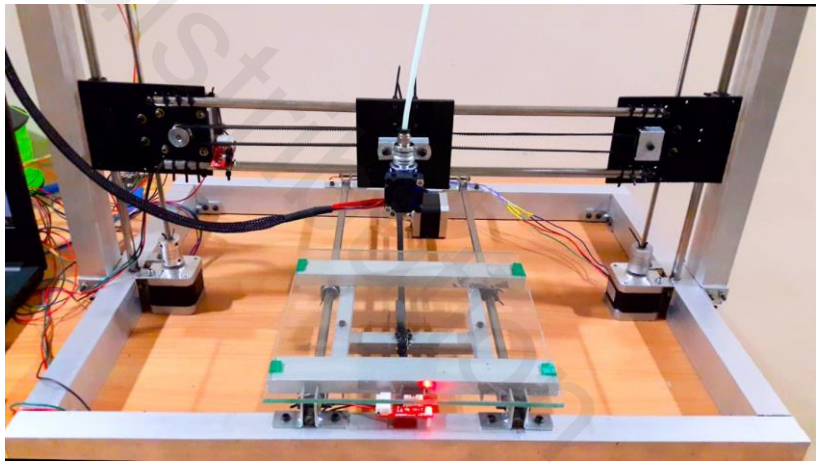




Examples of Arduino Applications

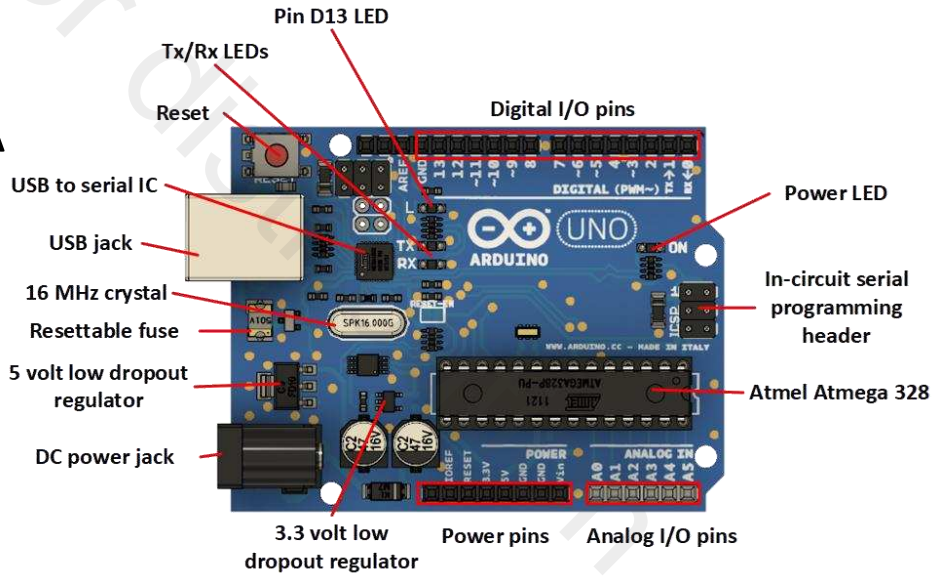
- Home automation system (https://tinyurl.com/ecasp-ha2018)
- RC Car Remote-controlled via Bluetooth (https://tinyurl.com/RCcarBT)
- RGB Interior Car Lighting (https://tinyurl.com/RGB-CarInterior-Arduino)
- Robotic Arm Controller, Omni Wheel Robot (https://tinyurl.com/ecasp-omni)
- Irrigation Controller (https://www.mysensors.org/build/irrigation)
- Homemade 3D Printer (https://all3dp.com/1/diy-arduino-3d-printer/)
- Heartbeat monitoring system (https://tinyurl.com/hbrsensor)





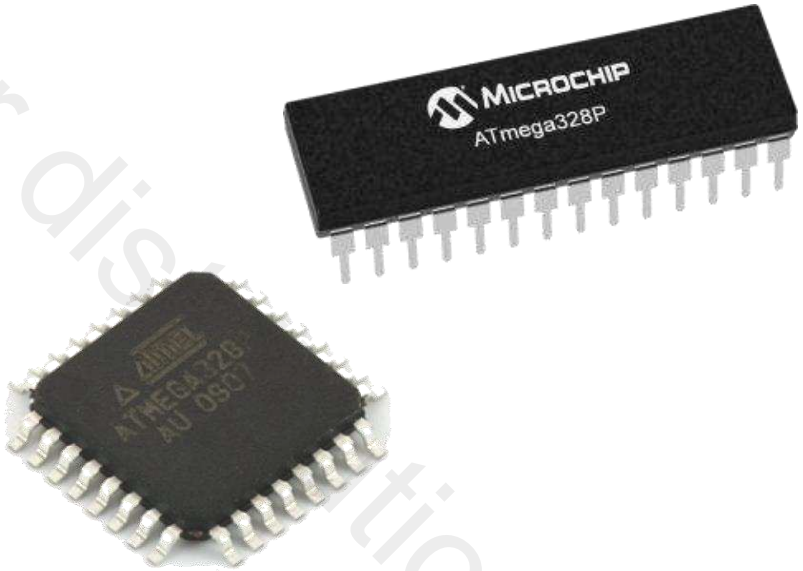
Arduino UNO Hardware Specification

- Microcontroller: ATmega328P
- Operating Voltage: 5V
- Input Voltage(recommended): 7-12V
- Input Voltage(limits): 6-20V
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Analog Input Pins: 6
- DC Current per I/O Pin: 20mA
- DC Current for 3.3V Pin: 50mA
- Weight: 25g (0.88 oz)

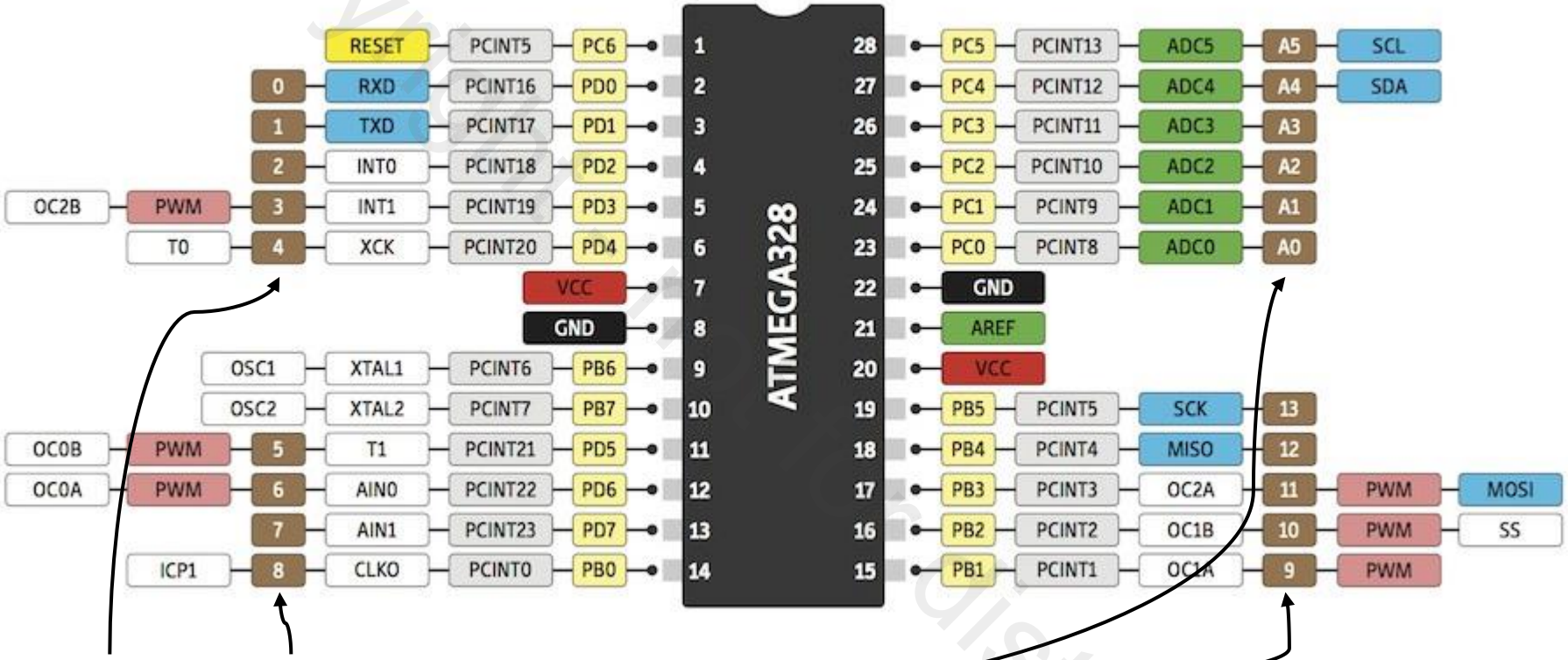


ATmega328P Microcontroller

- AVR 8-bit RISC architecture
- 20 MIPS at 20 MHz
- 32 KB In-system programming (ISP) flash memory with read-while-write capabilities
  - ISP: a technique where a programmable device is programmed after the device is placed in a circuit board (no need to remove from circuit to re-program)
  - 0.5KB used by bootloader
- 1 KB EEPROM
  - Long-term data
- 2 KB SRAM
  - Temporary values, stack, etc.
- 23 programmable I/O channels
- 3 timers/counters
- 6 PWM outputs

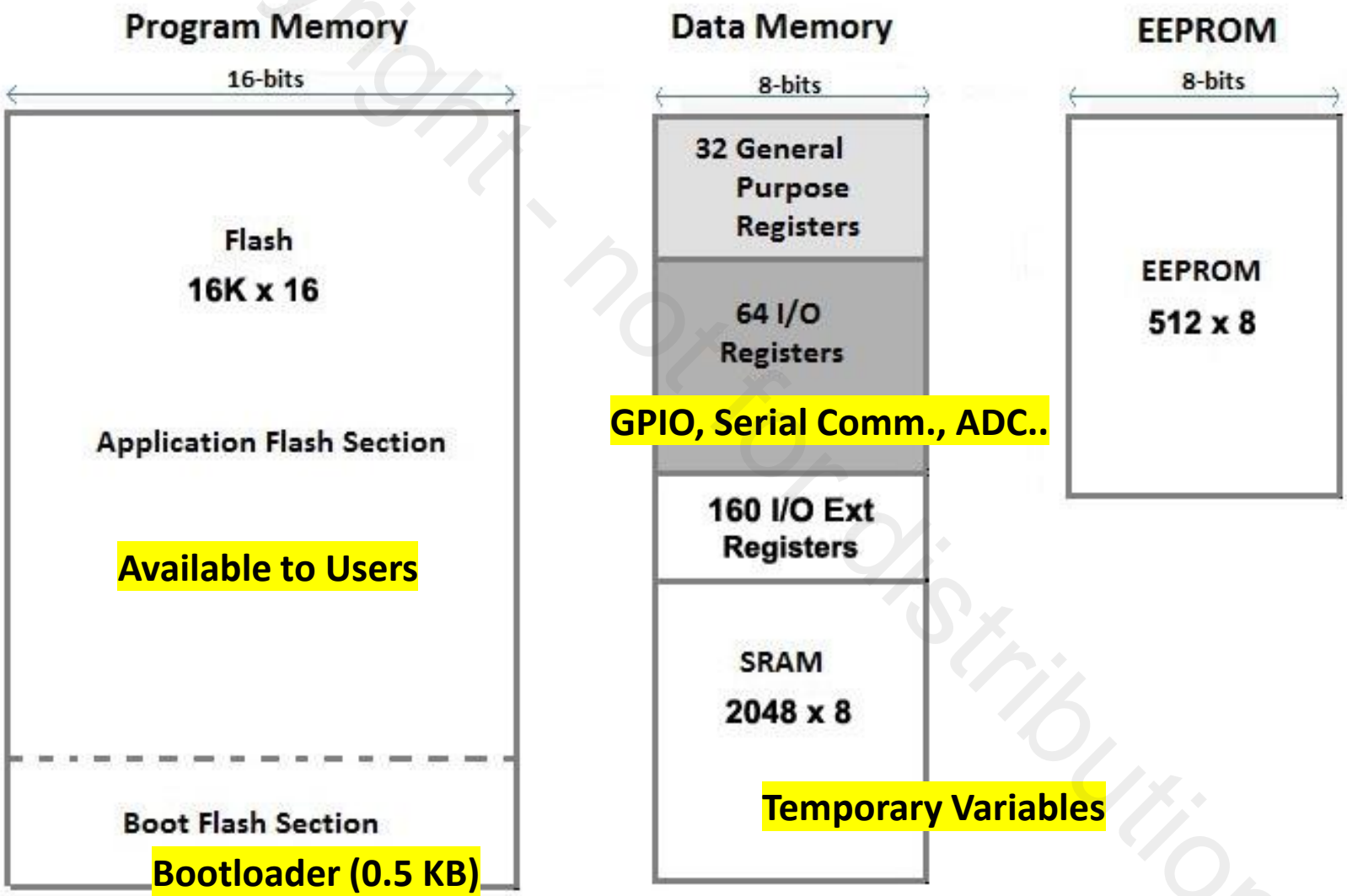


ATmega328P Microcontroller

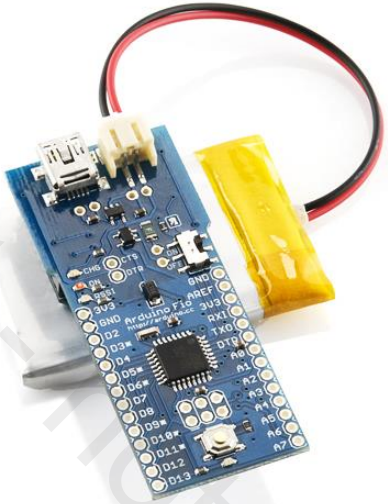


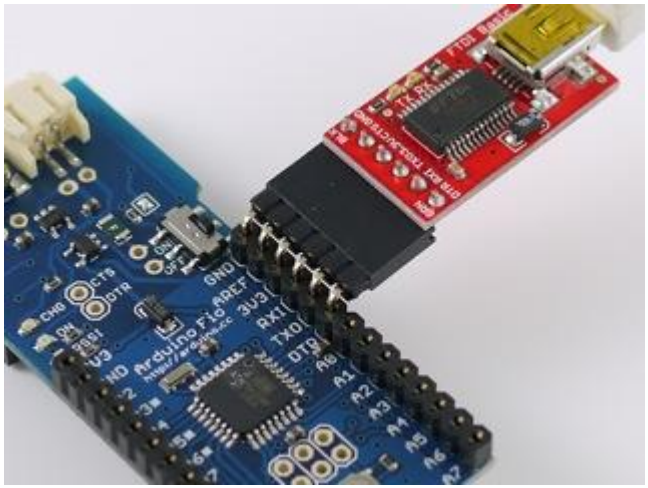
Arduino UNO Pin Numbers

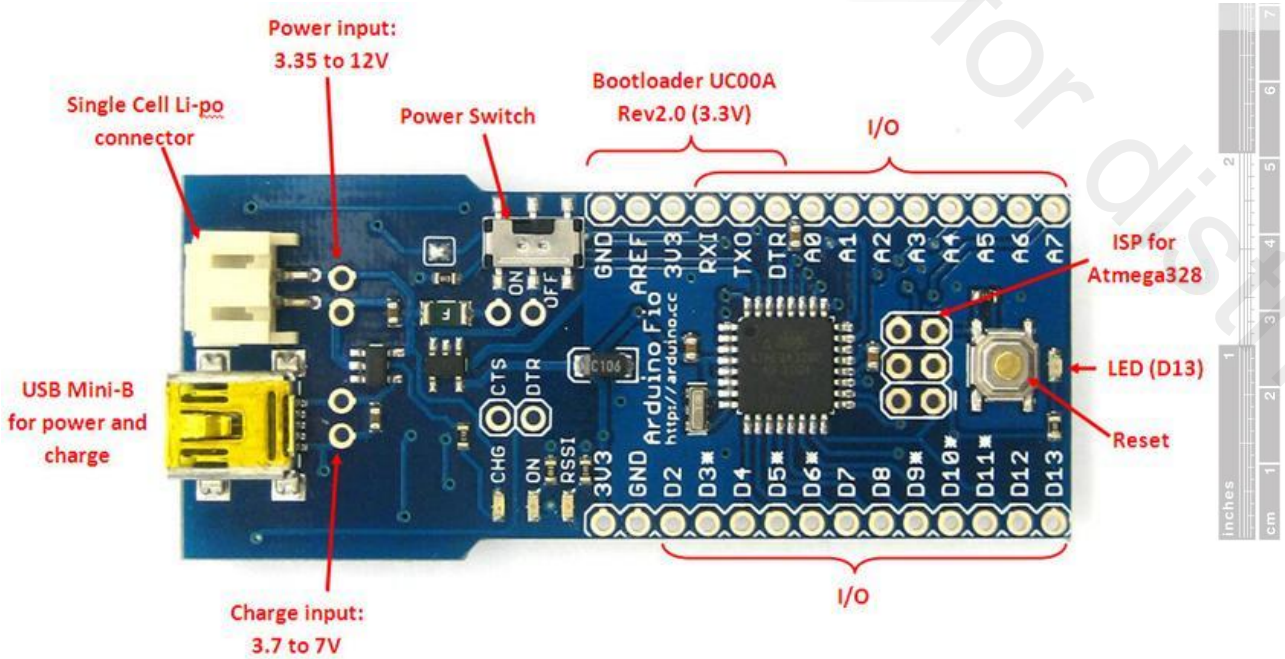
ATmega328P Memory Map

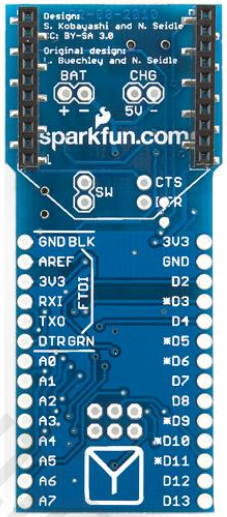


Arduino Fio







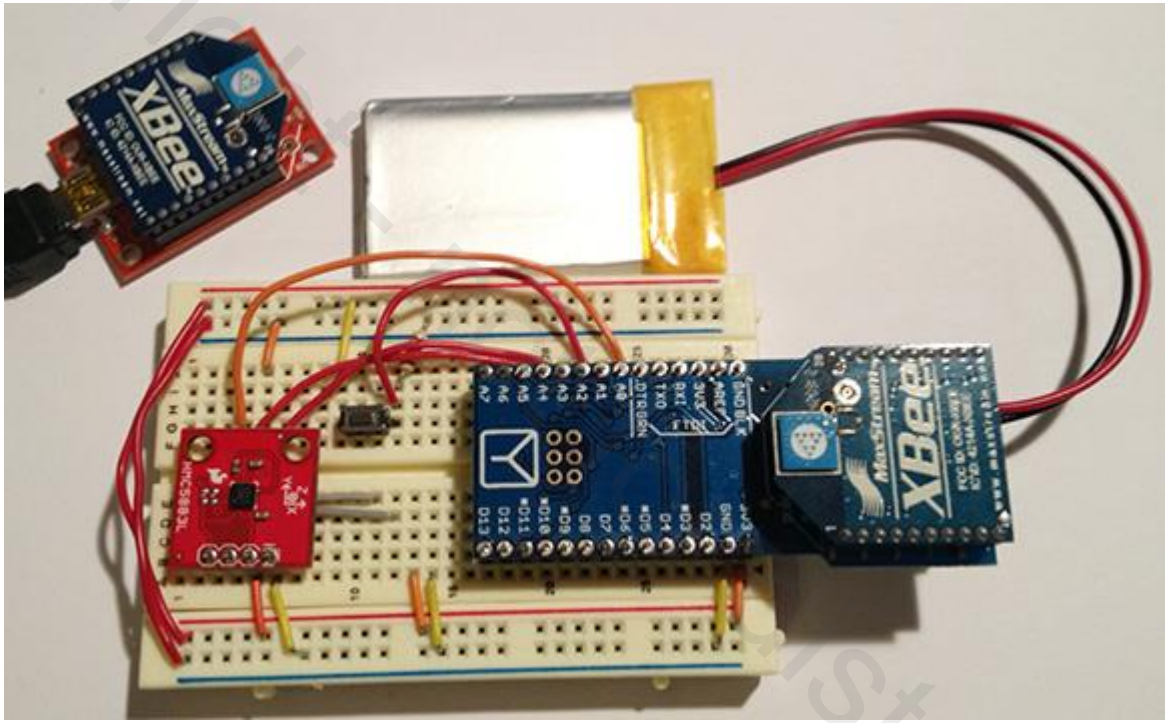


Arduino Fio Hardware Specification

- Microcontroller: ATmega328P at 8 MHz
- Operating Voltage: 3.3V
- Input Voltage: 3.35 – 12V
- Input Voltage for Charge: 3.7 – 7V
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Analog Input Pins: 8
- DC Current per I/O Pin: 40mA
- Flash Memory: 32KB (including 2KB for bootloader)
- Weight: 9g (0.32 oz)

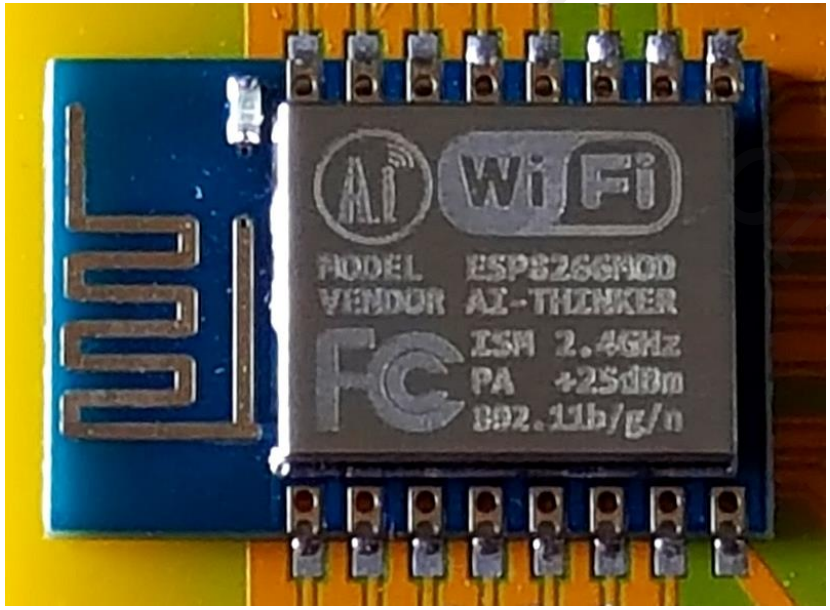
- *Require a separate programmer attached to this board*

Arduino Fio for XBee Communication



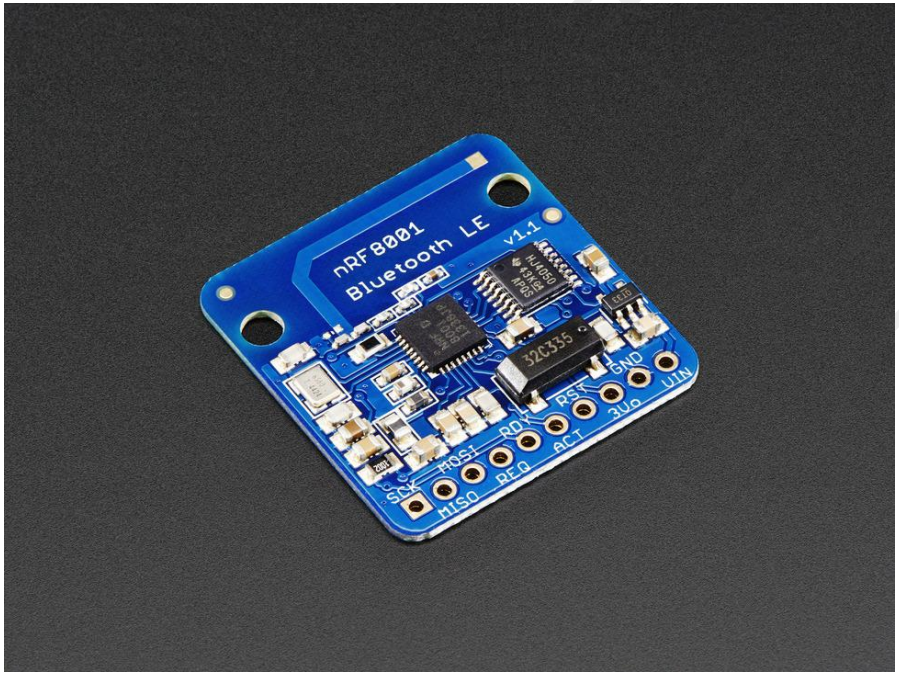
Simple Demo1: https://www.youtube.com/watch?v=CdAWVDfQrB4
Simple Demo2: https://www.youtube.com/watch?v=w3RBKo6HO3w

Add Wi-Fi to Arduino



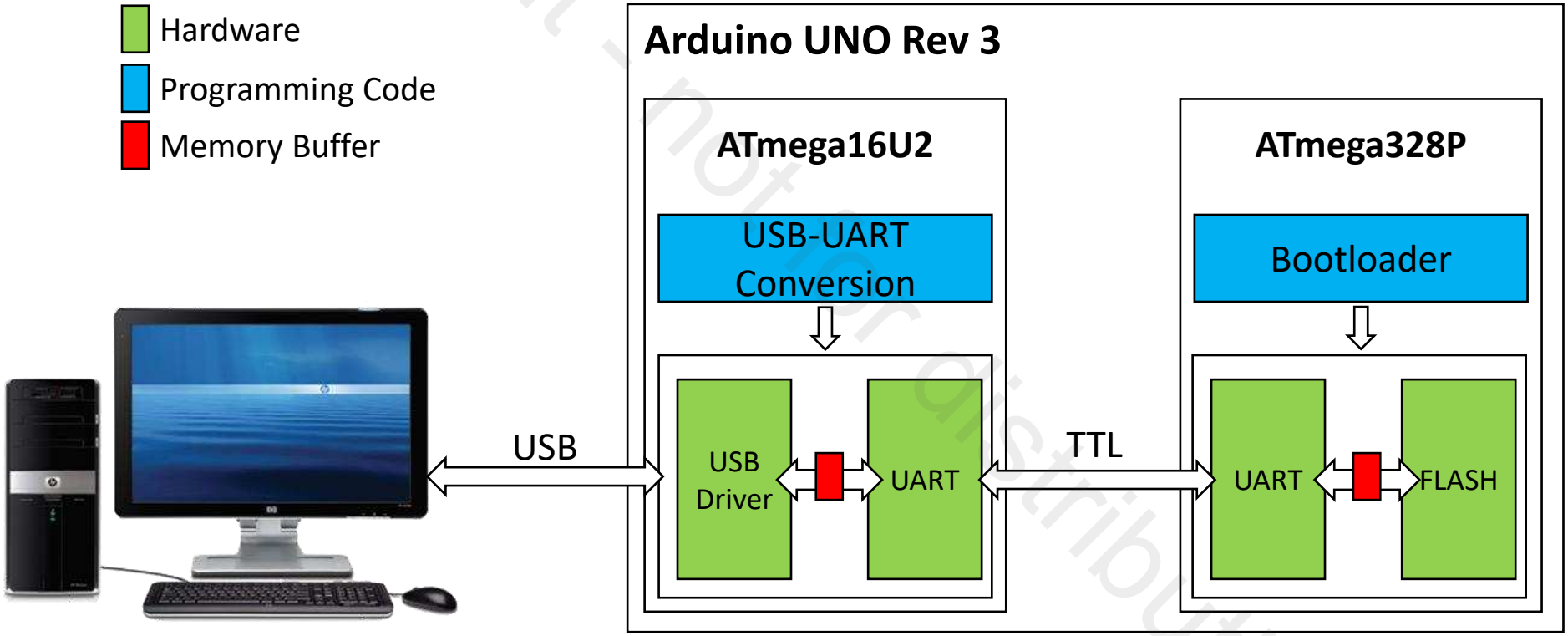
- Used in many IoT devices
- Cheap
- Easy to get (Amazon, Microcenter…)
- Easily can be added to any Arduino boards
- Shields are available as well

Add Bluetooth LE to Arduino

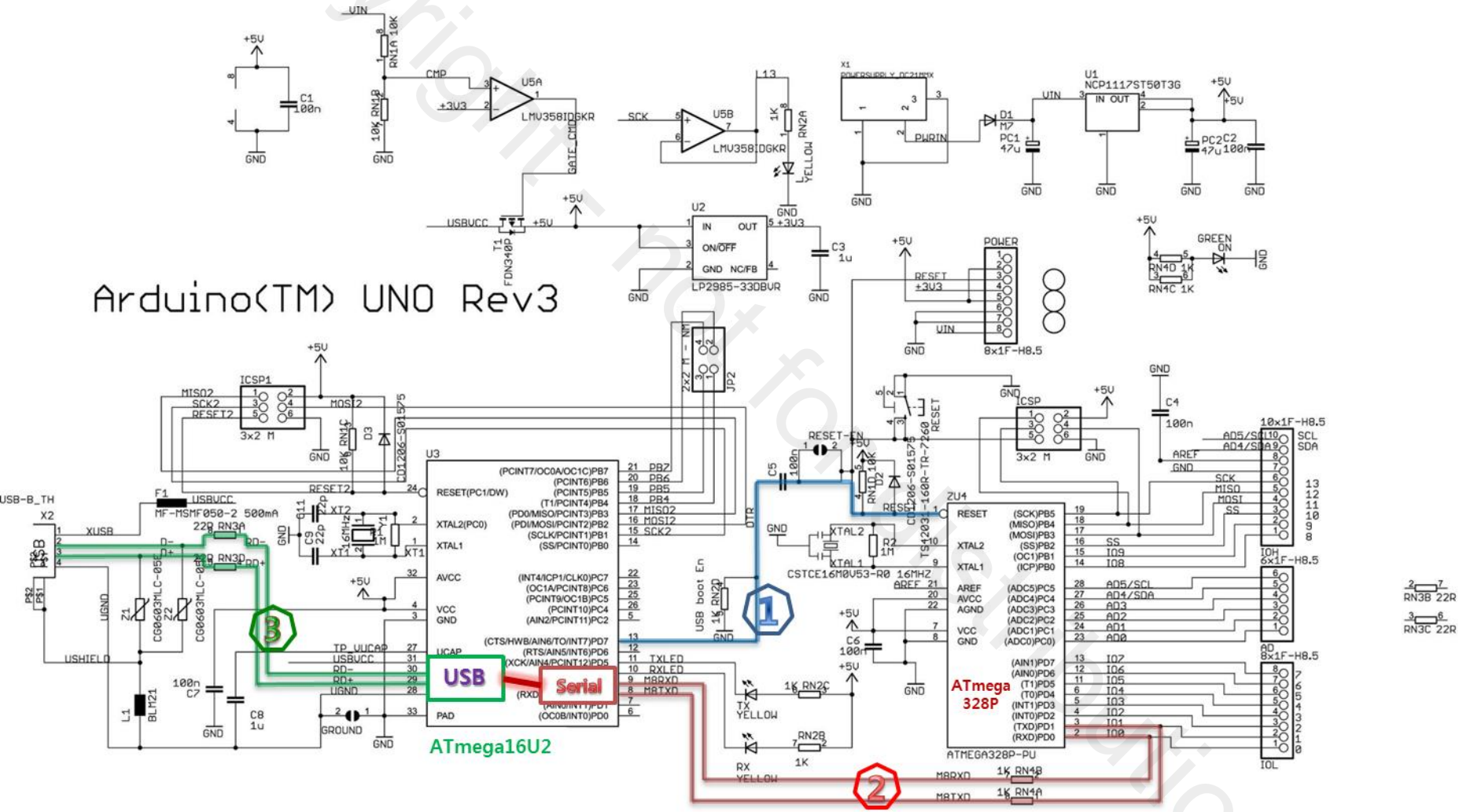


- nRF8001 Breakout
- Works between Arduino and any compatible iOS or Android (4.3+)
- Works by simulating a UART device, sending ASCII data
- Only $20
- You could go for a shield

Arduino UNO Upload Structure



Arduino UNO Upload Structure

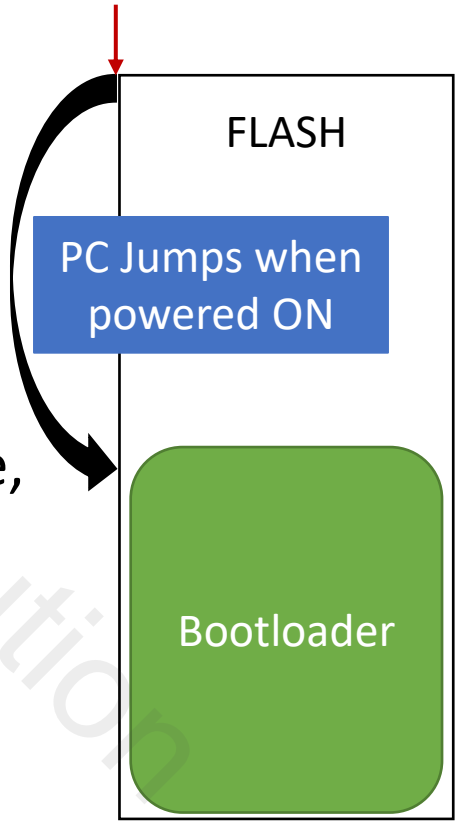


Arduino UNO Upload Structure

- USB to UART Conversion on ATmega16U2 for assembly code from PC
  - Assembly code generated from compiling the Arduino *Sketch*
- Bootloader writes incoming assembly codes from serial communication onto the flash memory
- If a RESET signal is invoked, bootloader decides to write to flash or execute existing program
- When Arduino IDE uploads the sketch…
  - DTR signal is transmitted to the target microcontroller, and bootloader is executed (See (1) from previous slide)
  - Data including codes transmitted via UART(TTL) connection (See (2) from previous slide)
- When upload is complete…
  - Bootloader jumps to the uploaded program address after reset

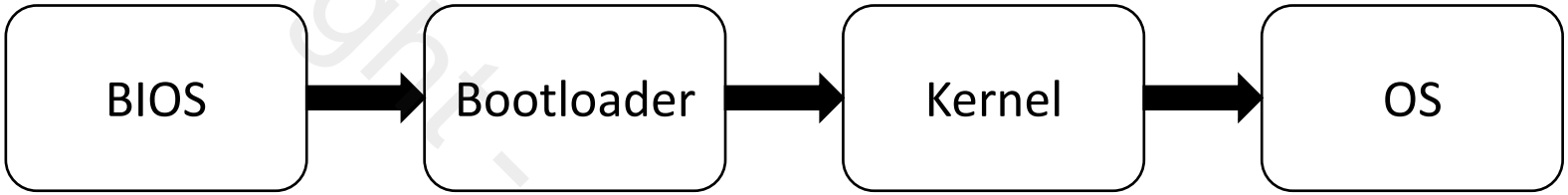
Bootloader

- A program that loads an operating system or the computer system when a computer is turned on
  - Bootloader is often called firmware mistakenly
  - Also called as boot manager or bootstrap loader
- Found in your laptop, desktop, smartphone, smartwatch, many embedded systems…
- A system software in the flash memory (ROM) that initializes all associated hardware, copies kernel to the memory (RAM), prepares the system for processing user commands
- Serial communication, interrupts, timer, console, memory, I/O peripherals, and others



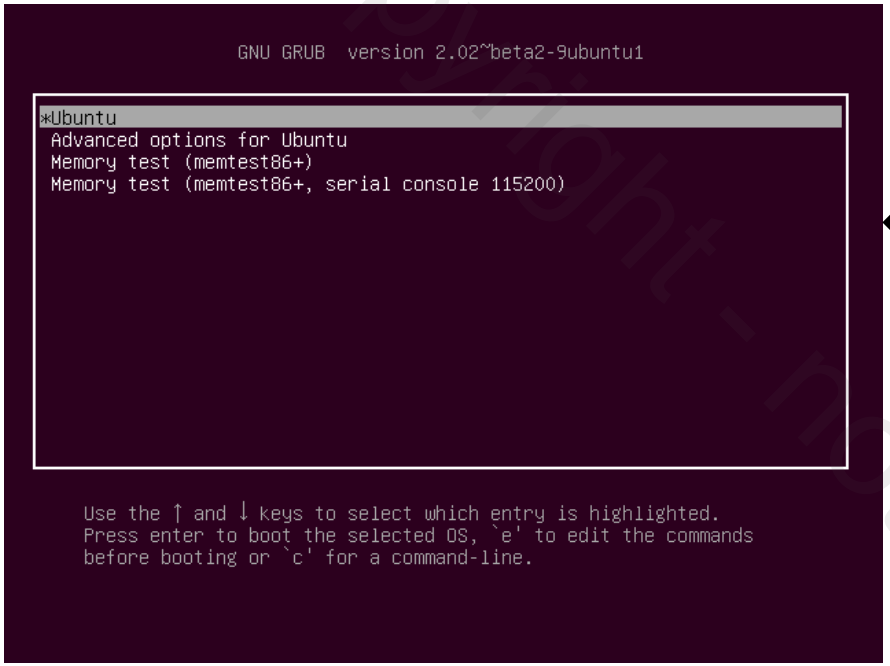
Bootloader

- Basic ‘booting’ sequence



  - BIOS (Basic Input Output System) loaded from ROM
  - BIOS calls bootloaders from all connected storage media sequentially
  - **Bootloader prepares all hardware** before loading kernel
  - After loading the kernel, framework is started, and UI is executed for the operating system
    - Kernel: where the commands are executed
    - Framework: delivers commands included in the program
    - UI: user interface, runs on top of kernel and framework
- Bootloader can be re-programmed by the user
  - For changing its original behavior due to changes on hardware/software, optimizations

Bootloader - Examples

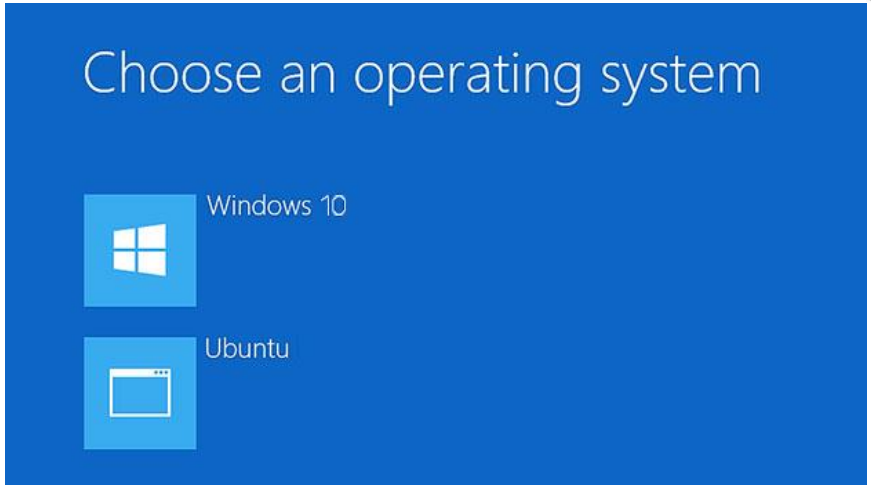


GNU GRUB Bootloader

Android Bootloader

Bootloader - Examples

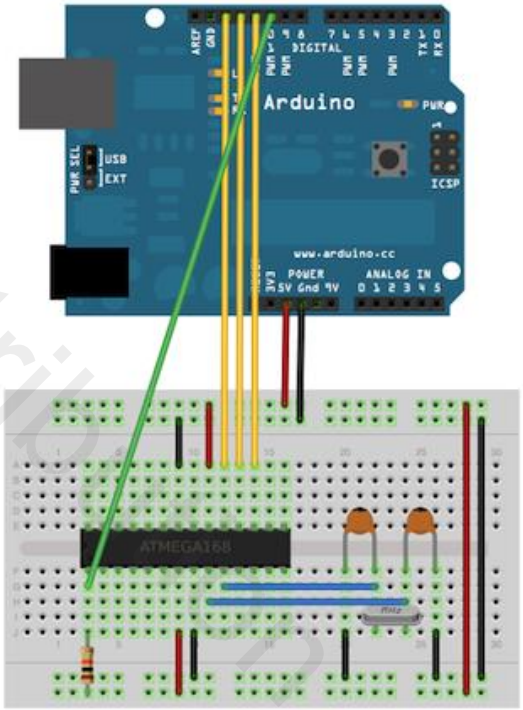
Windows 7 Boot Manager



Windows 10 Boot Manager

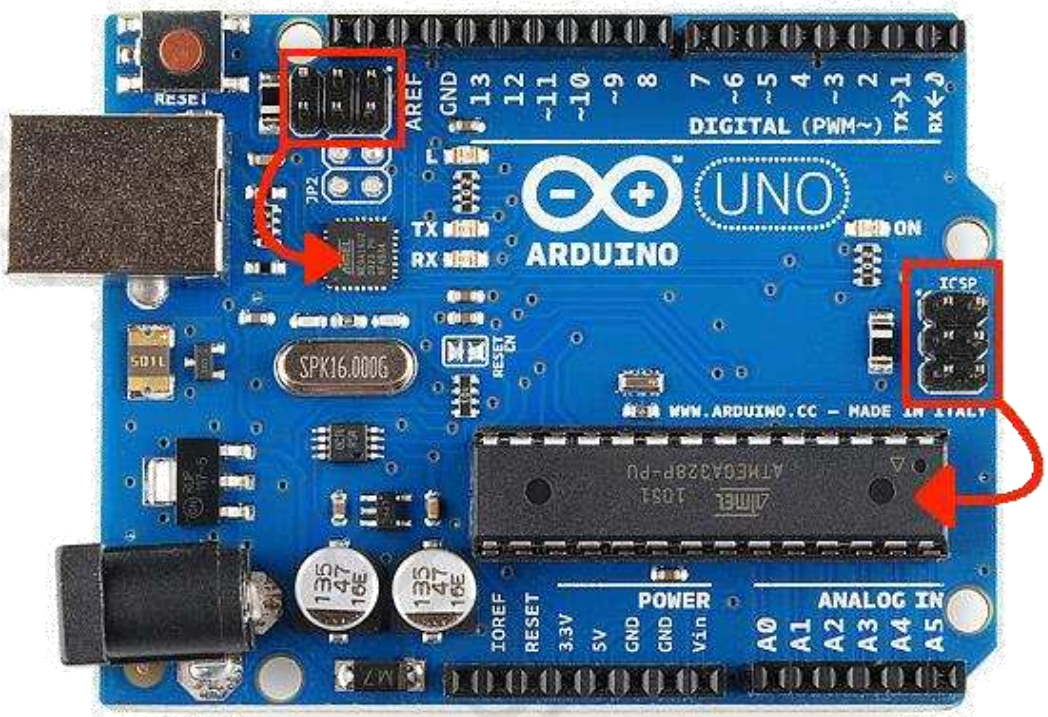
Arduino Bootloader

- Allows installing new firmware without the need of an external programmer
- Initializes the microcontroller for running existing program or for flashing new program to the memory
- May be reprogrammed by…
  - AVR Studio, Arduino IDE
  - Arduino board as ISP
- When do you need to reprogram?
  - Corrupted or needed upgrade
  - New chip that doesn't contain bootloader



Arduino Bootloader

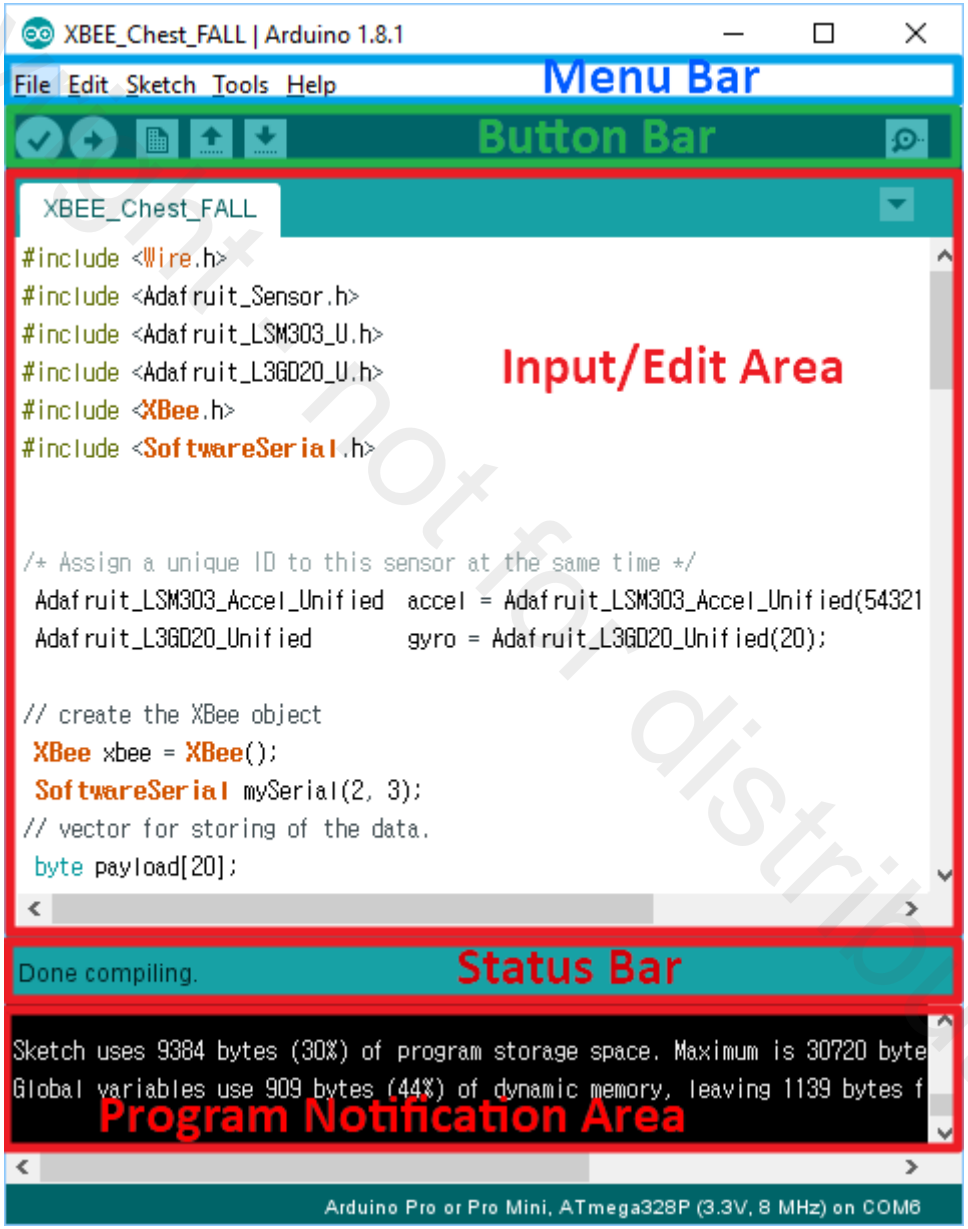
- Two ICSP (in-circuit serial programming) headers
- Reflashing firmware of USB interface chip (ATmega16u2), use ATmega16u2 ICSP headers
- Reflashing bootloader, use ATmega328 ICSP headers
- Pins are SPI Pins (MISO, MOSI, SCK), Vcc, GND and Reset



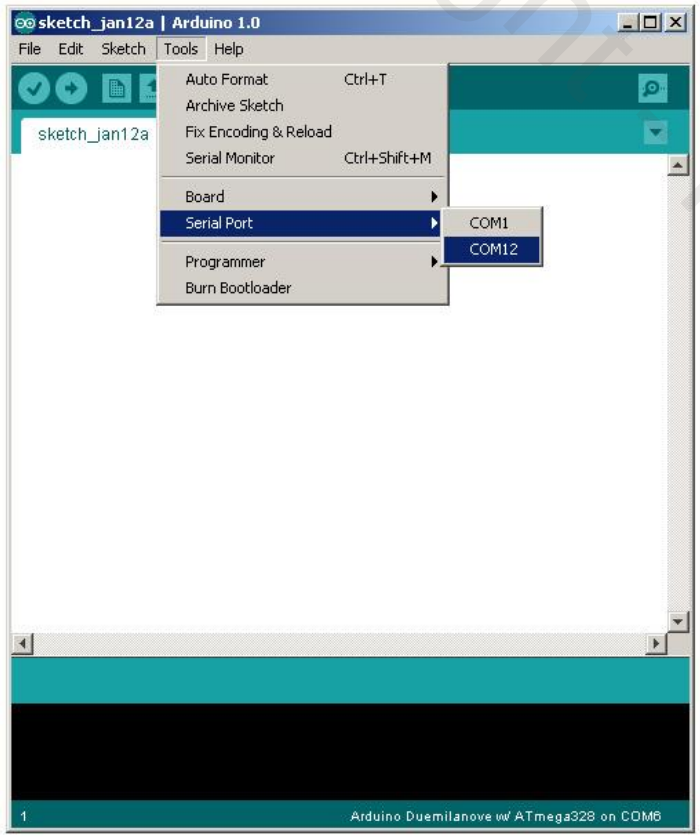
Lab 1
Traffic Light Controller Implementation using Arduino UNO

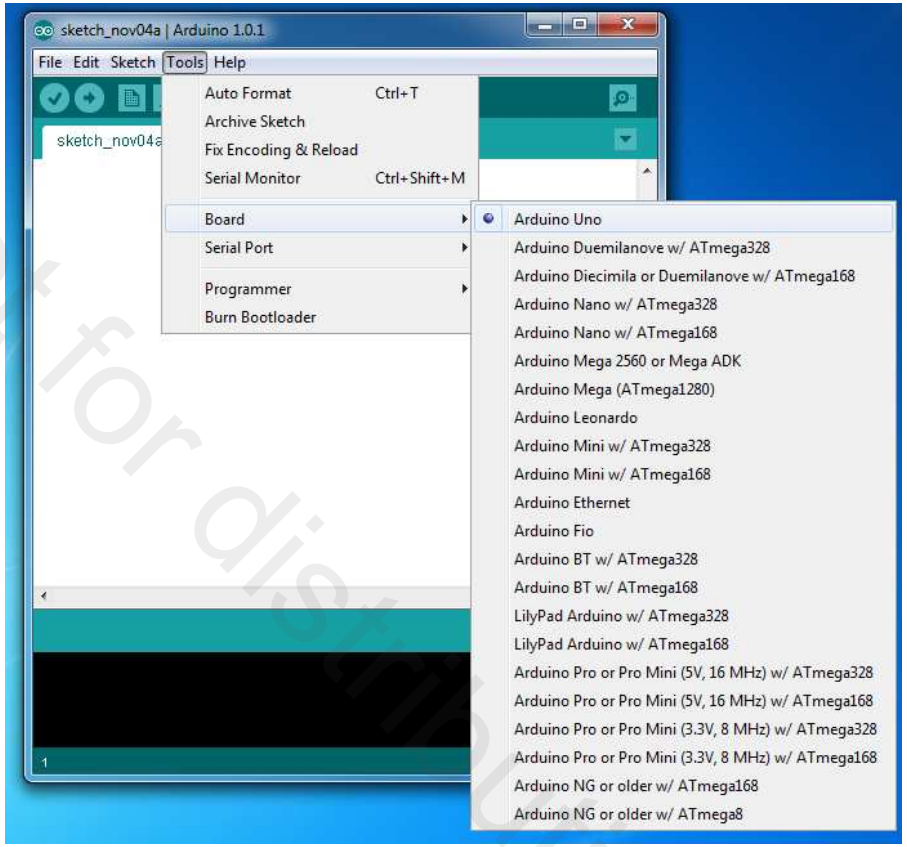
Lab explanation for Arduino
https://youtu.be/ZFMq9okTc3k

Arduino IDE



Select Serial Port and Board



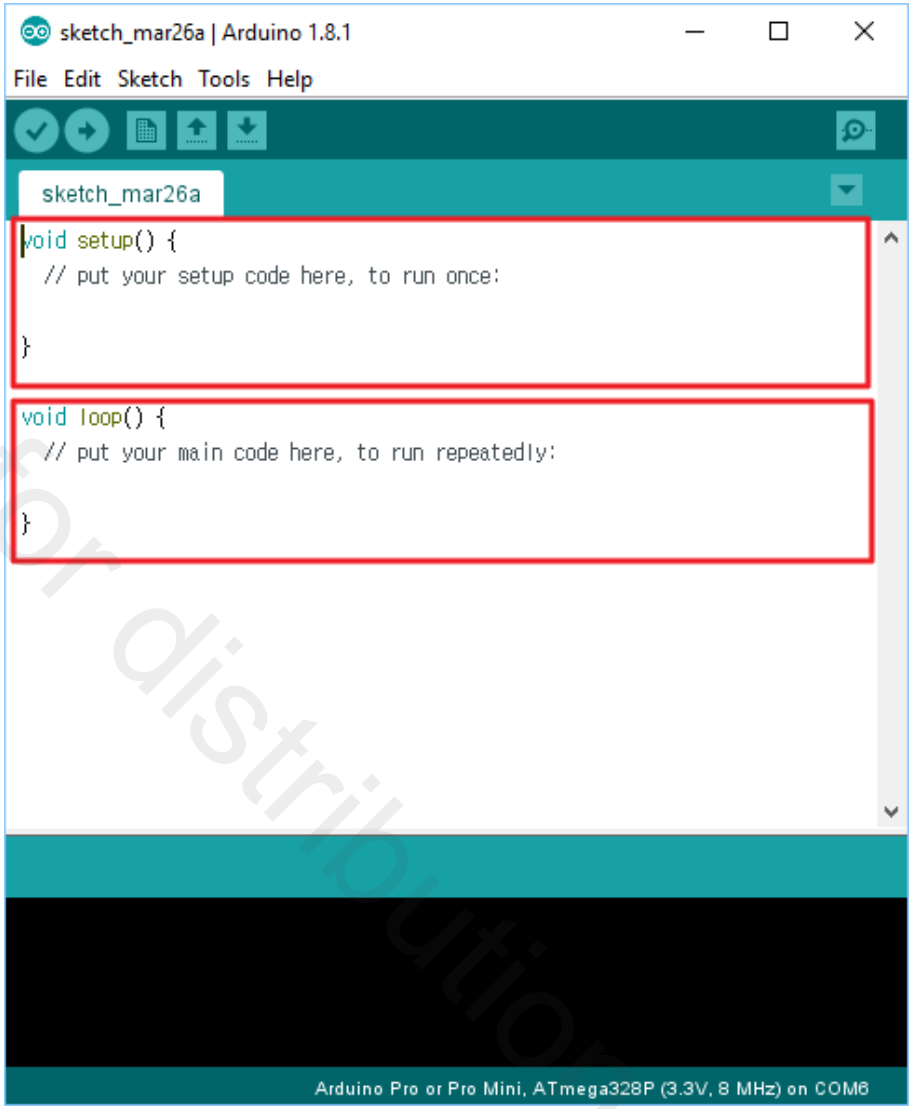


Arduino Programming Sequence

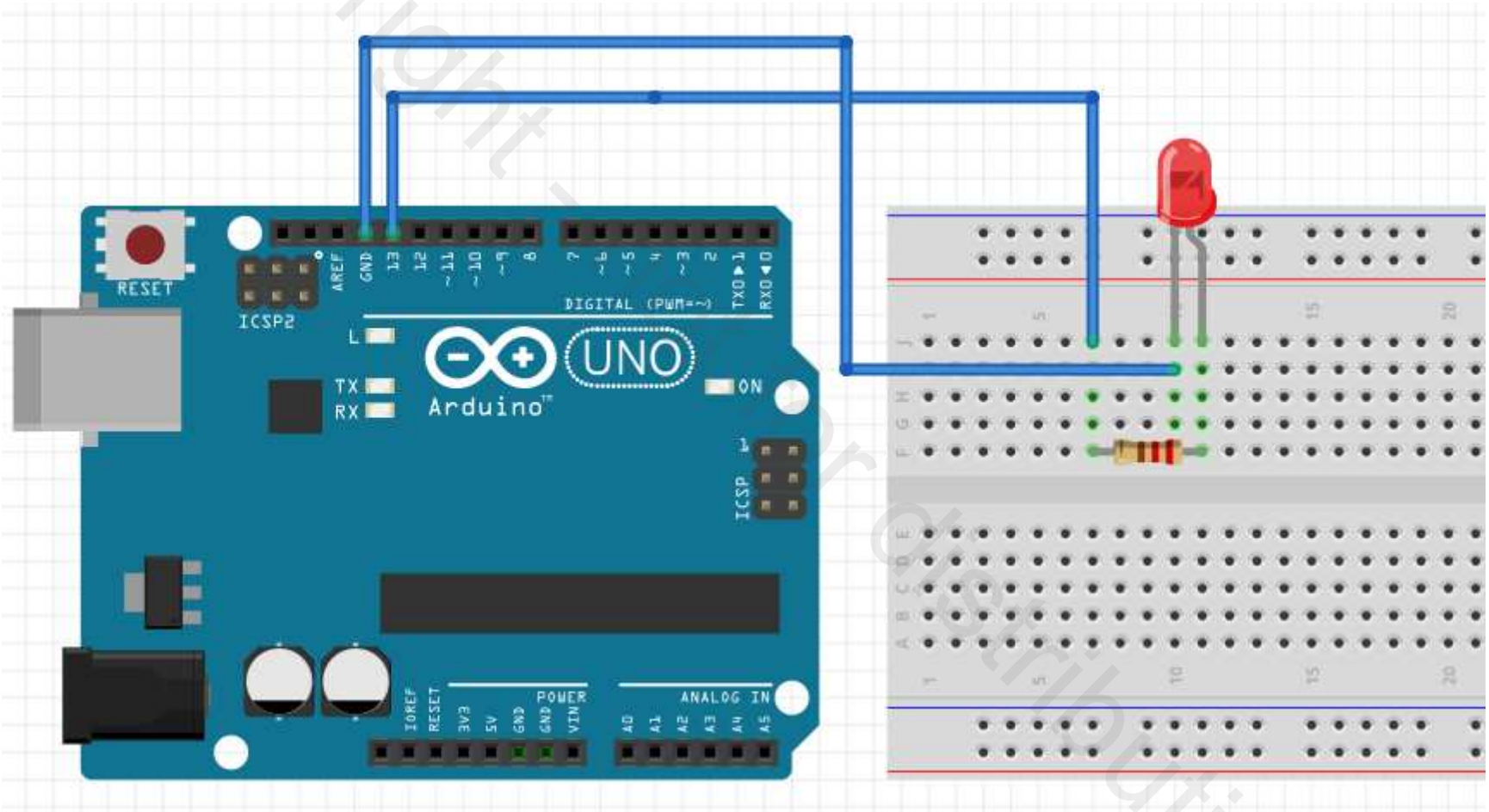
- Write your sketch/program
- Compile your sketch (to check for errors)
- Upload your sketch to the Arduino board via USB cable
- Verify your result on Arduino or Serial Monitor

Arduino Programming

- Case sensitive
- Statements are commands
- Must end with a semi-colon
- Comments follow a "//" or begin with /* and end with */
  - Same as C programming
- void setup()
  - Will be executed only when the program beings (or reset button is pressed)
  - Initializations for communications, variables, I/Os...
- void loop()
  - Will be executed repeatedly
  - Program behavior



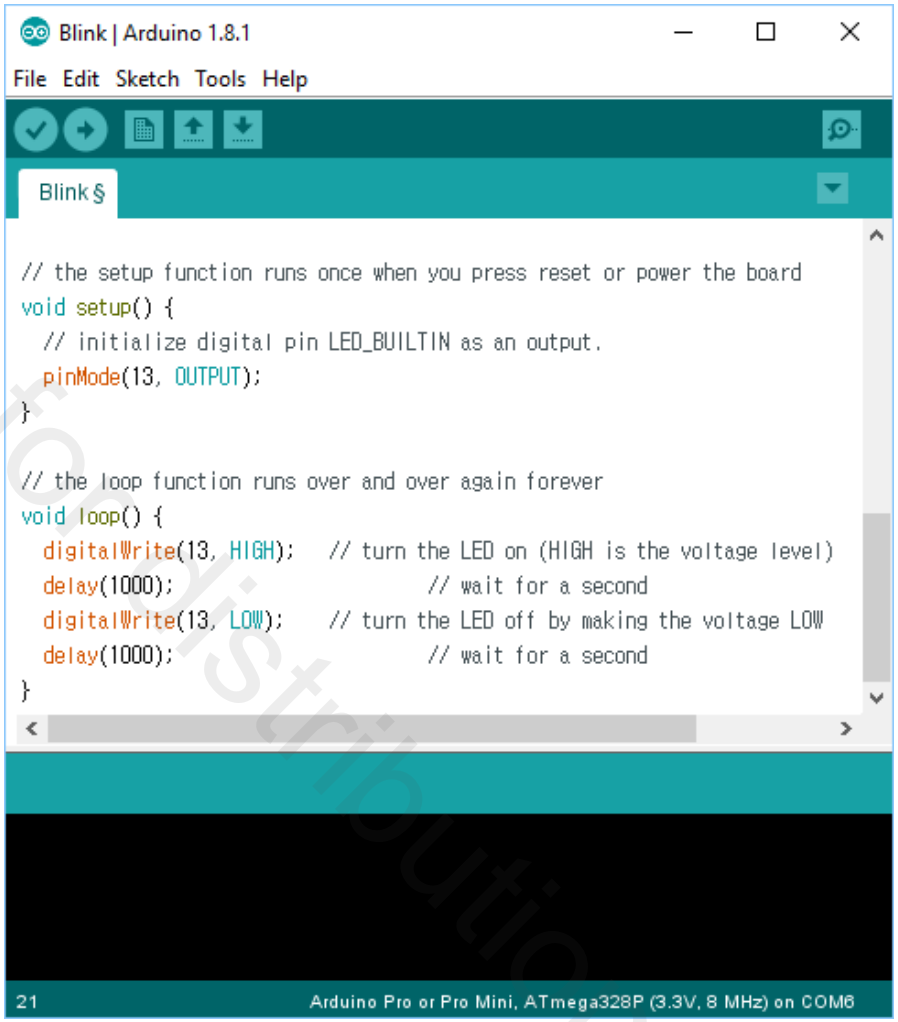
Programming Example – LED Blink



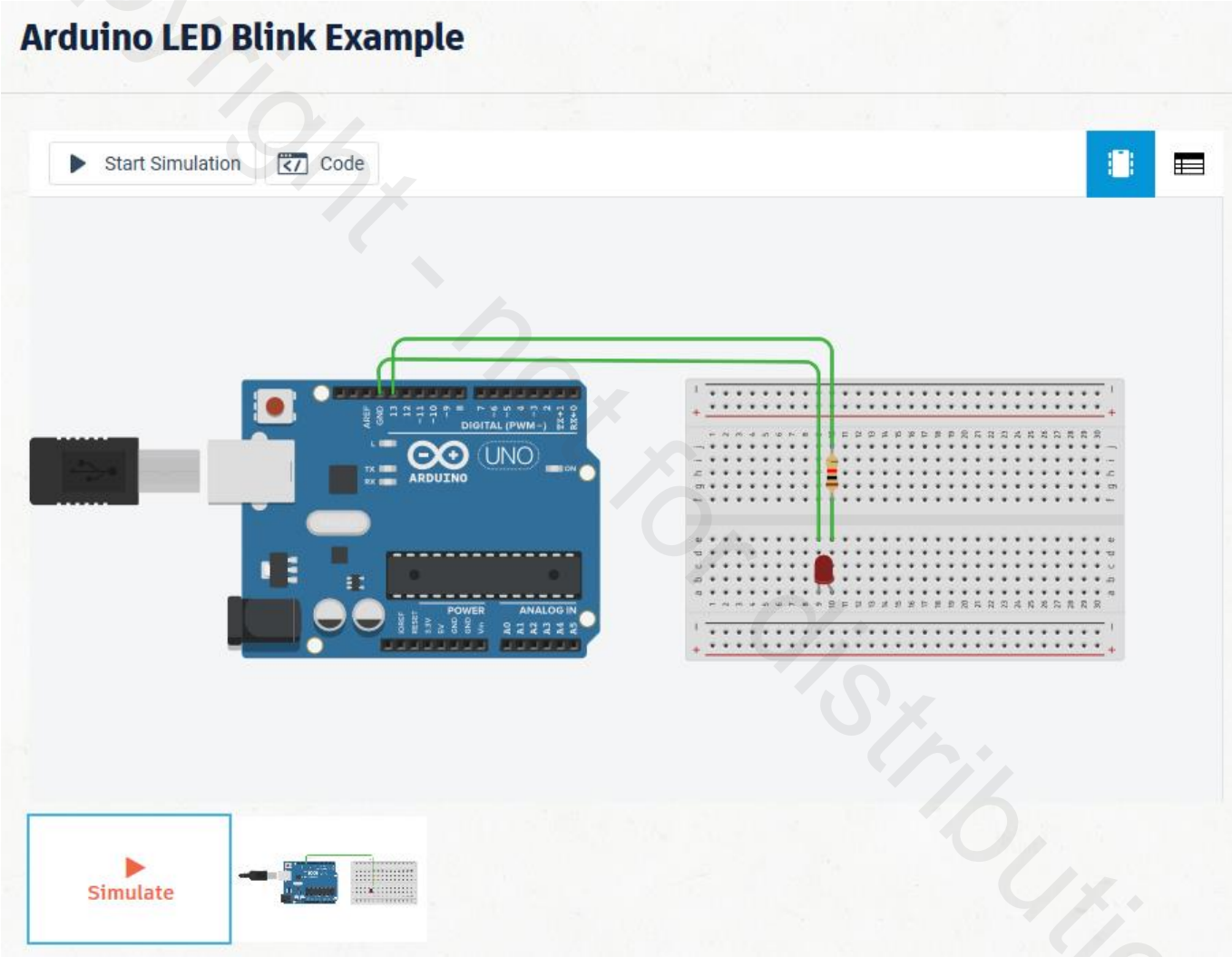
Programming Example – LED Blink

- pinMode(pin, mode)
  - pin: pin number on the board
  - mode: INPUT, OUTPUT, INPUT_PULLUP
- digitalWrite(pin, value)
  - pin: pin number on the board
  - Value: HIGH or LOW
- delay(value)
  - value: milliseconds
- Initializes Pin #13 as OUTPUT
- Writes HIGH to Pin #13
- Waits for 1000ms (1s)
- Write LOW to Pin #13
- Waits for 1000ms (1s)

- LED will blink every 1 second



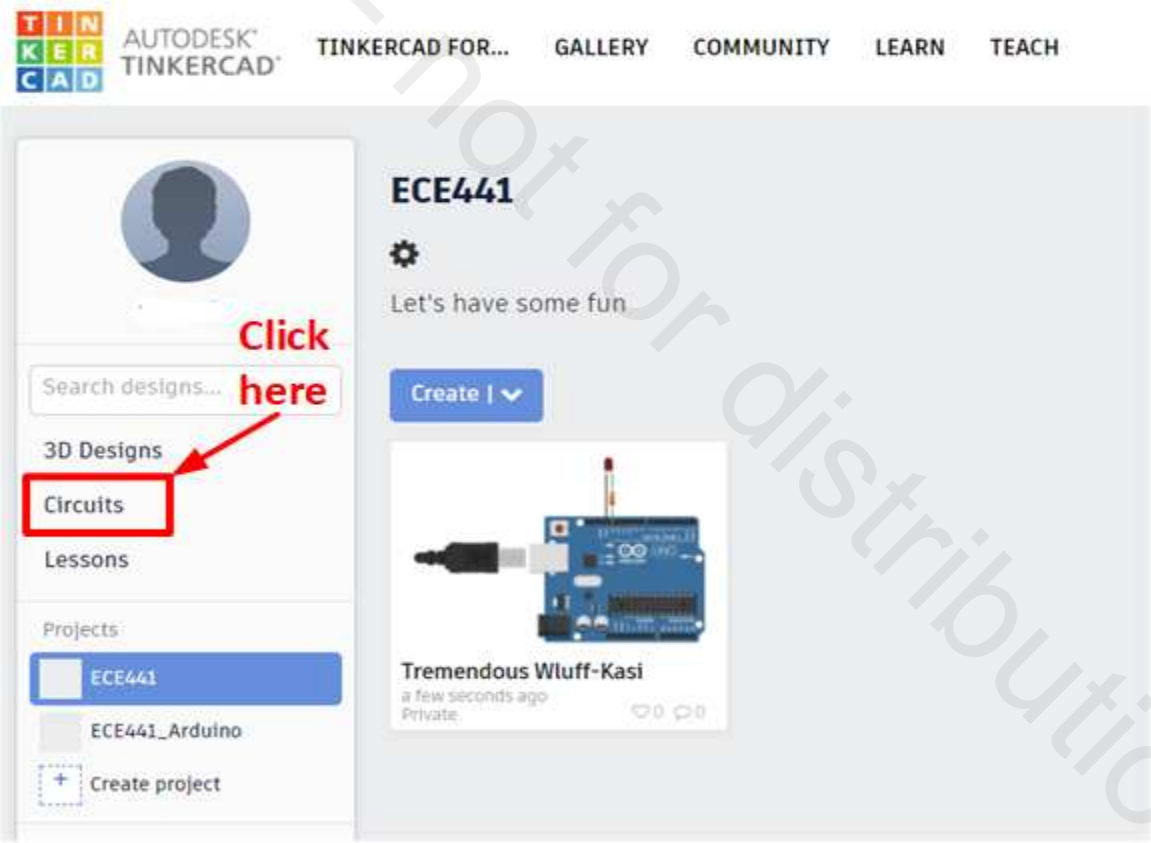
Programming Example – LED Blink



https://www.tinkercad.com/things/9bv1Y2v2Ea8

TinkerCAD

- Go to the website: https://www.tinkercad.com
- Register for Autodesk account using your Illinois Tech email address
- Click "**Circuits**" on the left panel

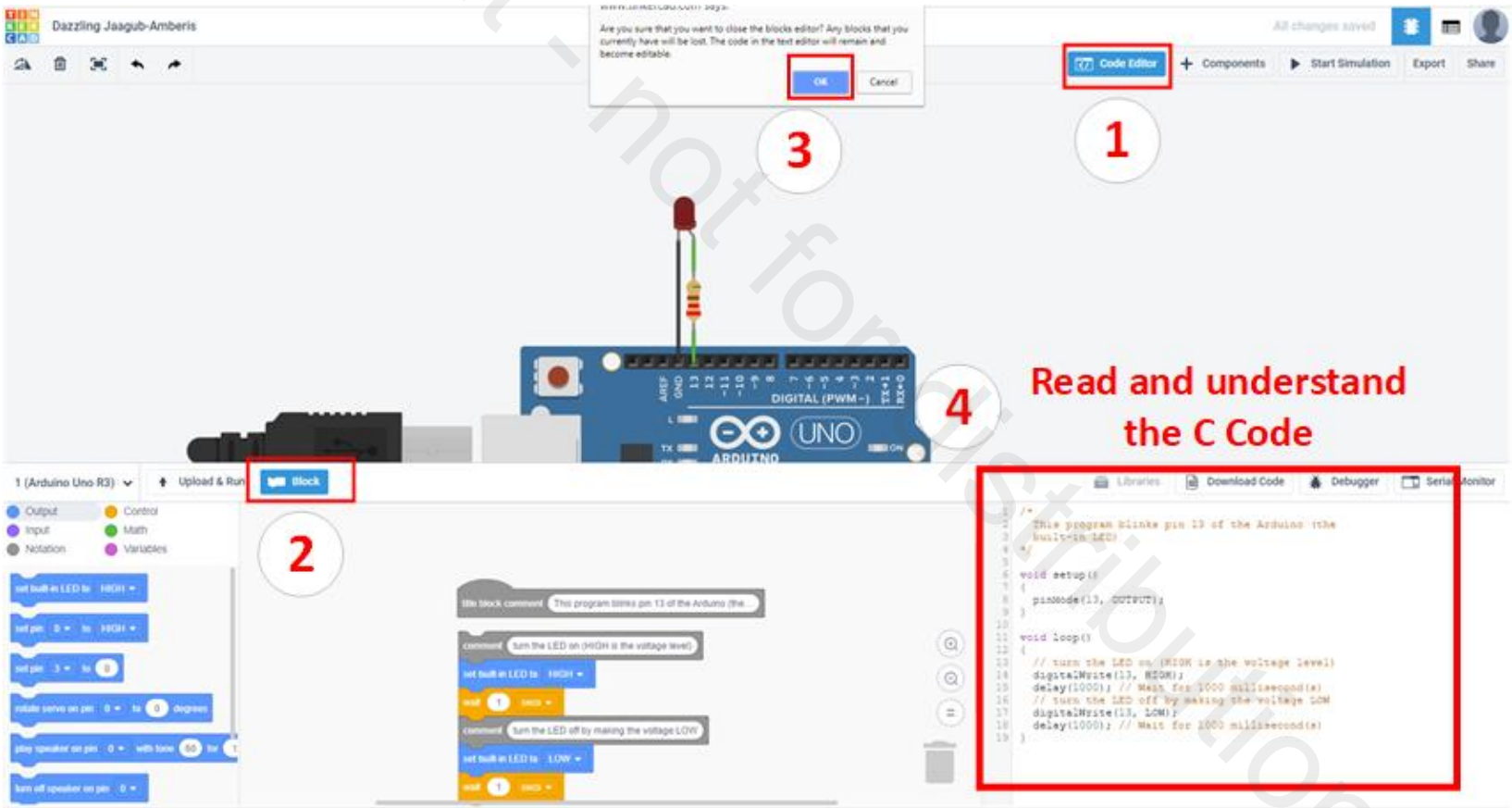


TinkerCAD

- Click "**Create**" -> "**Circuit**" -> "**+ Component**" and add "**Arduino Blink**" from the "**Starters**" Tab.

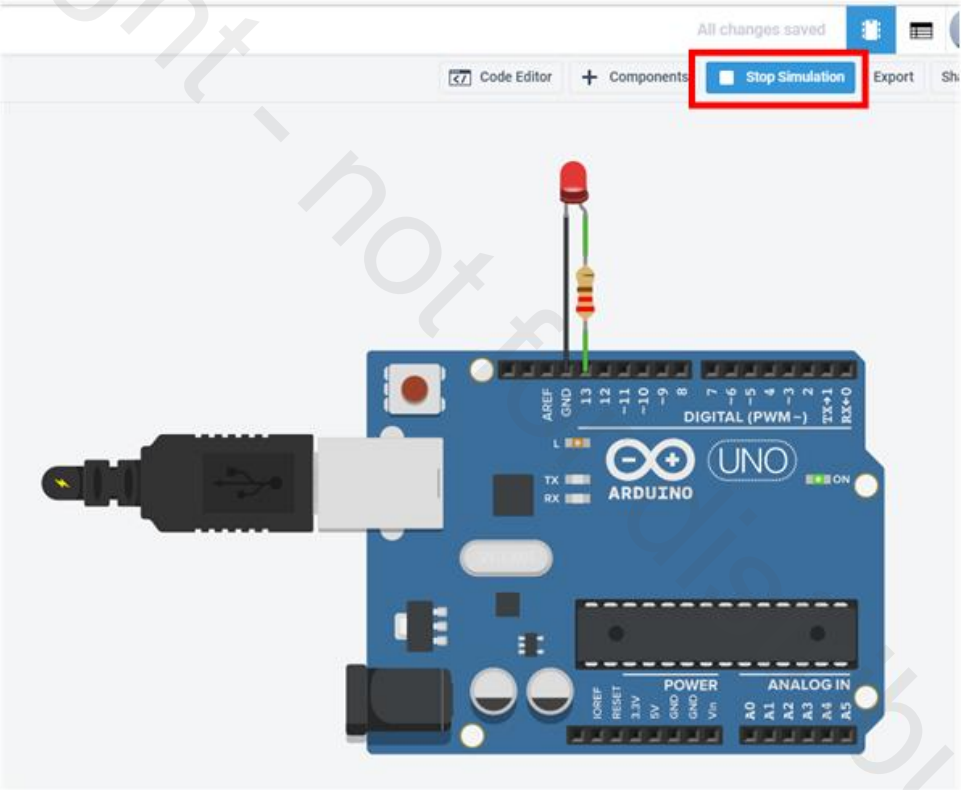
TinkerCAD

- Go to "**Code Editor"** and disable "**Block**" function since we don't want to program it using the block diagram. Read the C code to gain a better understanding of the program flow.



TinkerCAD

- Click **“Start Simulation”** and observe the result.



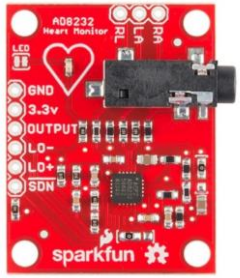
I/Os in Arduino UNO

- Analog I/O
  - Built-in ADC (Analog-to-Digital Converter) allows analog inputs into a digital value (Analog IN Pin #A0 to Pin #A5)
  - Absence of DAC (Digital-to-Analog Converter), PWM (Pulse-Width Modulator) can achieve some of the functions of an analog output
  - Libraries associated with each analog input and output
- Digital I/Os
  - GPIO. Input and output of HIGH and LOW values
  - Three (digital) serial communication protocols on Arduino UNO to interface with peripherals
  - Digital I/O Pin #0 to #13 on Arduino UNO
  - UART, SPI, I^2C depending on the type of the peripherals
  - Libraries associated with each communication protocols
  - Initialize, data transmit, read/write from/to device

Analog I/O

- analogRead(*pin*)
  - Converts value of the voltage on an analog input pin to digital
  - Returns a digital value from 0 to 1023 (Resolution: 10-bit ADC)
  - Reference is 5V on most Arduinos including Arduino UNO
  - 7V on Arduino Mini, Nano
  - 15V on Arduino Mega
- analogWrite(*pin, value*)
  - Output a PWM square wave signal to Pins #3, #5, #6, #9, #10, #11
  - Value from 0 to 255 (Resolution: 8-bit) compared against an 8-bit counter value
  - Output pin will be HIGH from counts 0 to *value*, LOW from counts *value+1* to 255
  - Frequency of PWM signal on most pins approximately 490 Hz
  - Pins #5, #6 have 980 Hz on UNO, Pins #3, #11 on Leonardo

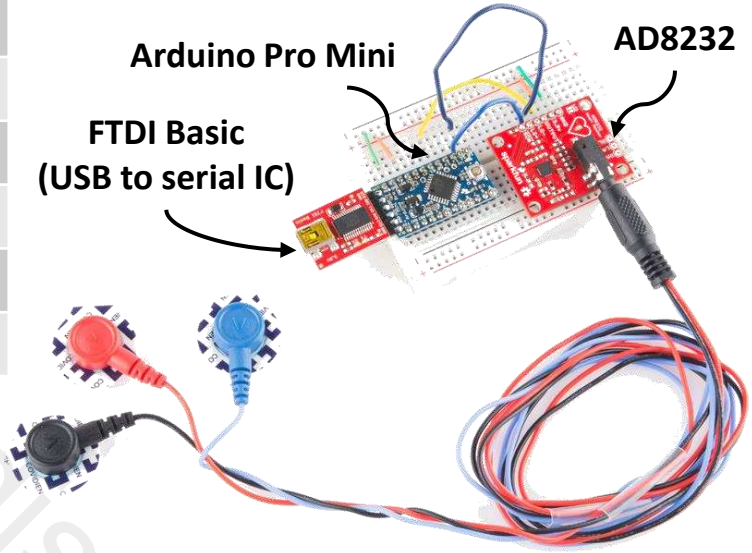
Analog I/O - Example



AD8232

- AD8232 Heart Monitor (ECG Sensor)

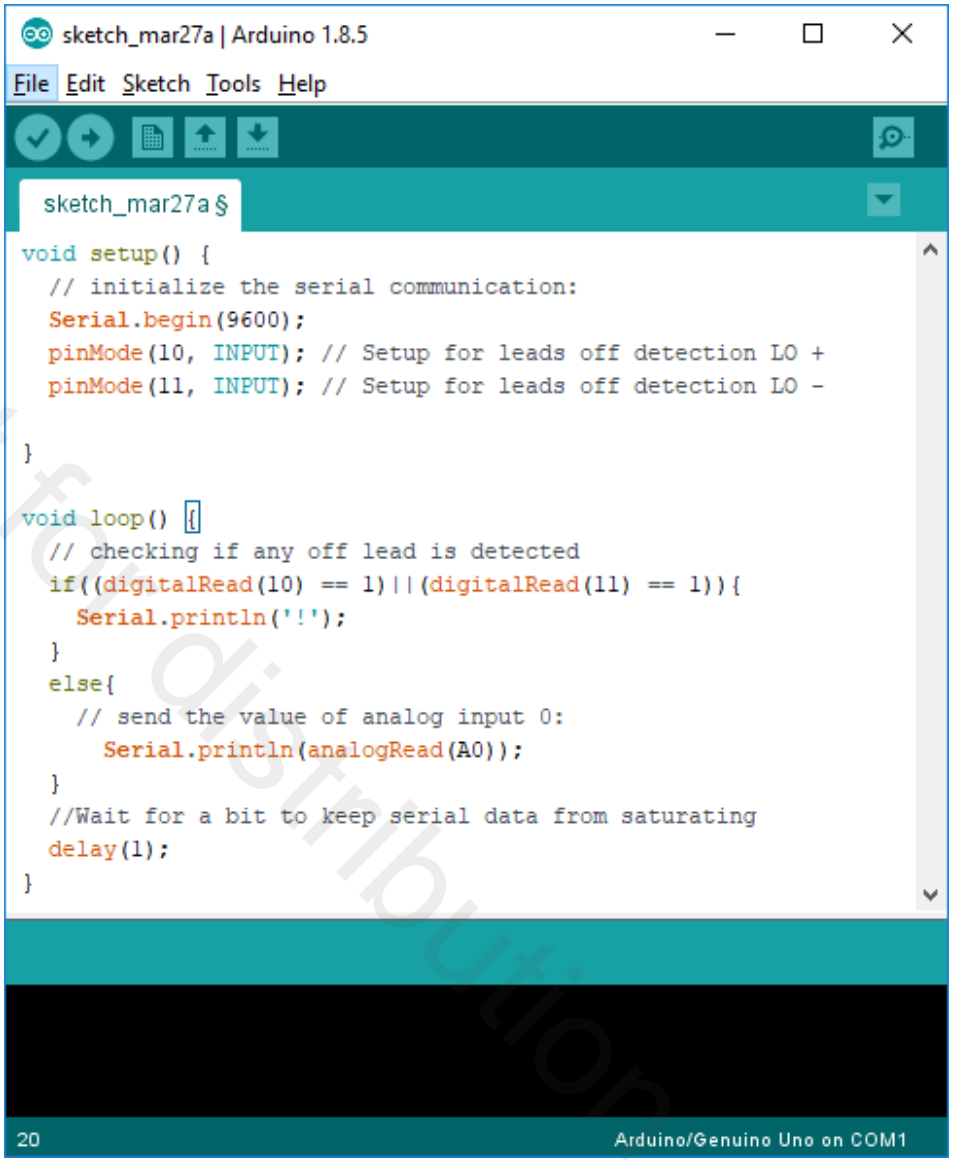
Board Label	Pin Function	Arduino Connection
GND	Ground	**GND**
3.3V	3.3V Power Supply	**3.3V**
OUTPUT	Output Signal	**A0**
LO-	Leads-off Detect -	**11**
LO+	Leads-off Detect +	**10**
SDN	Shutdown	**Not used**



- Generates ECG (Electrocardiogram) signal from 3 lead electrode pads
- Output is an analog signal, Arduino board will convert to value between 0 to 1024

Analog I/O - Example

- Class Serial in Arduino
  - Digital Pin #0 (Rx) #1 (Tx), USB connection
  - Use built-in serial monitor to communicate with the Arduino board (debugging…)
  - begin(*speed*): sets data rate in bps (baud)
  - println(*value*): prints data to the serial port
- digitalRead(*pin*)
  - pin: pin number on the board
  - Return value is either 0 (LOW) or 1 (HIGH)
- analogRead(*pin*)
  - pin: pin number on the board
  - Return value (0 – 1023)
- Initializes serial port communication at data rate of 9600bps
- Sets Pin #10, #11 as INPUT
- If any off lead is detected, print '!'
- Read in ECG signal data from A0, and print to the serial port




```
void setup() {
  // initialize the serial communication:
  Serial.begin(9600);
  pinMode(10, INPUT); // Setup for leads off detection LO +
  pinMode(11, INPUT); // Setup for leads off detection LO -

}

void loop() {
  // checking if any off lead is detected
  if((digitalRead(10) == 1)||(digitalRead(11) == 1)){
    Serial.println('!');
  }
  else{
    // send the value of analog input 0:
      Serial.println(analogRead(A0));
  }
  //Wait for a bit to keep serial data from saturating
  delay(1);
}
```

Digital I/O - UART

- Higher level software libraries provided for developers
  - No need to worry about start bit, stop bit, optional parity bit
  - Libraries
- *Serial*, *SoftwareSerial* libraries for UART implementation
  - *Serial*: Digital Pin #0 (Rx) #1 (Tx), USB connection, use built-in serial monitor to communicate with the Arduino board (debugging…)
  - *SoftwareSerial*: developed to allow serial communication on other digital pins, using software to replicate the functionality, similar to Serial but can choose Rx and Tx pins
- Designed for communication between two devices at a time
  - No method of differentiating multiple transmissions on the same line
- Half-duplex connection between two devices
  - Walkie-talkie-like communication

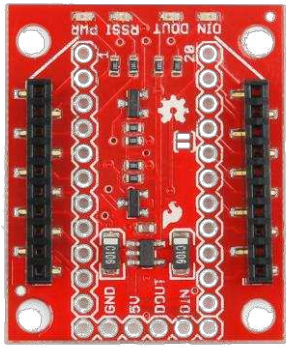
Digital I/O – UART implementation

Serial/Software Serial Method	Purpose	Code Example	Explanation
Constructor (SoftwareSerial ONLY)	Define GPIO pins as UART Rx and Tx	SoftwareSerial mySerial(2, 3);	Defines a serial connection with Rx on Pin #2, Tx on Pin #3
begin	Define data rate for serial connection	mySerial.begin(9600);	Communication on “mySerial” port will occur at 9600 baud
print/println	Write byte data into human-readable characters over serial connection	mySerial.println(“Hello World”);	Writes bytes equivalent to *Hello World*
write	Write raw byte data over the serial connection	mySerial.write(29);	Writes byte with value 29
read	Read data from the serial connection	mySerial.read();	Reads from serial connection

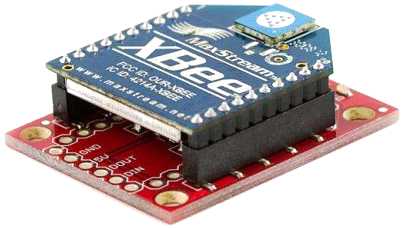
Digital I/O – UART Example

- XBee + XBee Explorer Regulated

Board Label	Pin Function	Arduino Connection
GND	Ground	**GND**
5V	5V Power Supply	**5V**
DOUT	Data Output (Tx)	**2**
DIN	Data Input (Rx)	**3**



XBee Explorer Regulated



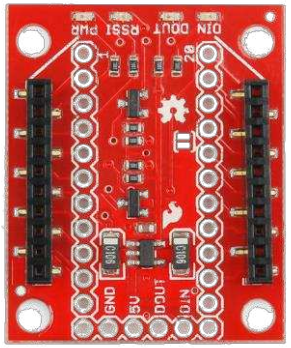
XBee Module

- XBee module is for ZigBee connection
- XBee Explorer Regulated translates 5V serial signals to 3.3V
- DOUT(Tx) and DIN(Rx) values are HIGH or LOW
- XBee's Tx is Arduino's Rx, XBee's Rx is Arduino's Tx
- Use *SoftwareSerial* library for data communication between Arduino board and XBee module
- Use *XBee* library for initializing, data packaging, setting receiver information…

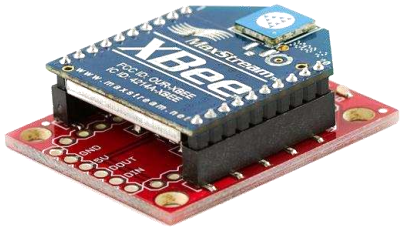
Digital I/O – UART Example

- XBee + XBee Explorer Regulated

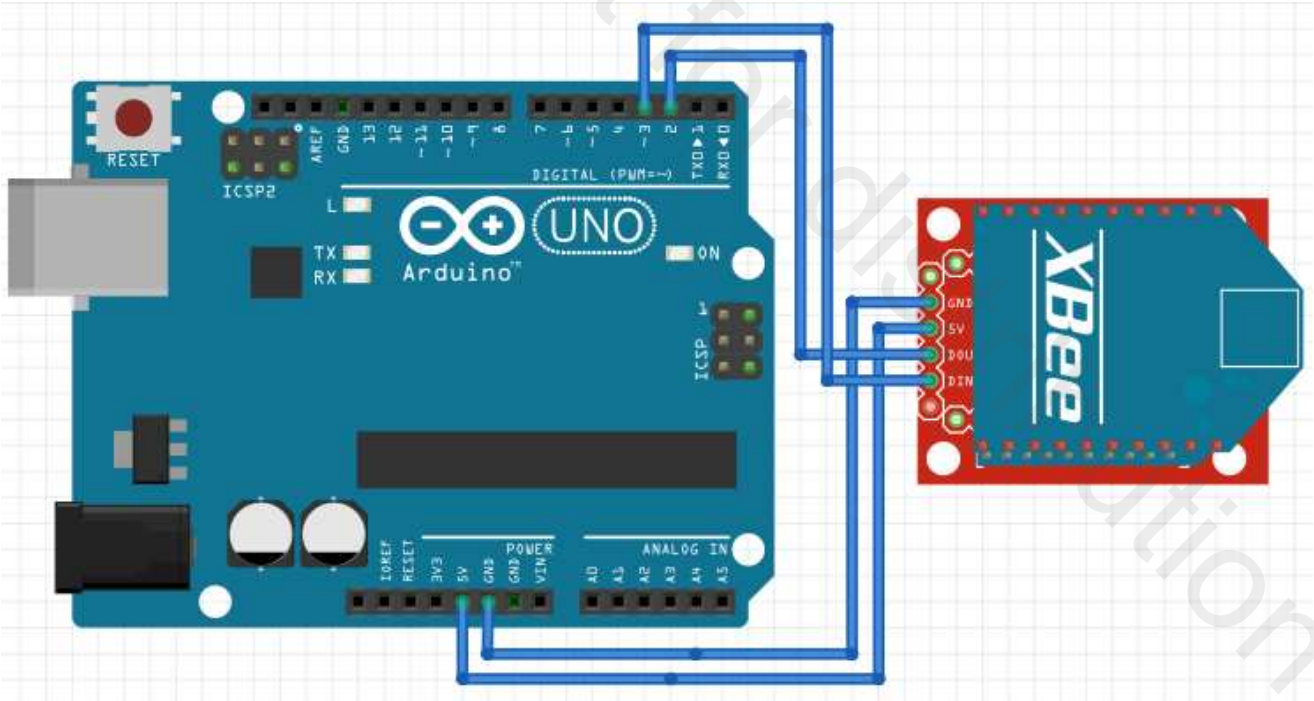
Board Label	Pin Function	Arduino Connection
GND	Ground	**GND**
5V	5V Power Supply	**5V**
DOUT	Data Output (Tx)	**2**
DIN	Data Input (Rx)	**3**



XBee Explorer Regulated



XBee Module



Digital I/O – UART Example

- Include required headers
  - XBee.h for XBee module behaviors
  - SoftwareSerial.h for SoftwareSerial
- Initialize pin #2, #3 to *mySerial*
- Necessary initialization for XBee
  - Set receiver's address
  - Prepare Tx data for ZigBee
- Set correct data rate between XBee and Arduino board
- Set mySerial as for sending and receiving packets over ZigBee
- payload[] contains data to be Tx
- send() function transmits packet to Tx including receiver's address, payload size, payload…

XBEE_Chest_FALL | Arduino 1.8.1

File Edit Sketch Tools Help

XBEE_Chest_FALL §

```
#include <XBee.h>
#include <SoftwareSerial.h>

 XBee xbee = XBee(); // create the XBee object
 SoftwareSerial mySerial(2, 3); // set Pin 2 as Rx, Pin 3 as Tx
 byte payload[20]; // payload for zigbee data transfer

// XBee Initializations
XBeeAddress64 addr64 = XBeeAddress64(0x0013A200,0x41241F14);
ZBTxRequest zbTx = ZBTxRequest(addr64, payload, sizeof(payload));
ZBTxStatusResponse txStatus = ZBTxStatusResponse();

void setup(void)
{
  mySerial.begin(57600); // set mySerial at 57600 baud
  xbee.setSerial(mySerial); // set mySerial for xbee
}
void loop(void)
{
  // pack payload[]
  // payload[2] = ....
  // payload[10] = ....

  // SENDS the information to the addressed XBEE module
  xbee.send(zbTx);
}
```

Done compiling.

```
Sketch uses 3720 bytes (12%) of program storage space. Maximum is 30720 byte
Global variables use 482 bytes (23%) of dynamic memory, leaving 1566 bytes f
```

25 Arduino Pro or Pro Mini, ATmega328P (3.3V, 8 MHz) on COM6