MICROCONTROLLERS

OVERVIEW

Microcontrollers

- Uses
- Types

Schematics and Features

ADC

Programming Microcontrollers

Interaction Design with Microcontrollers

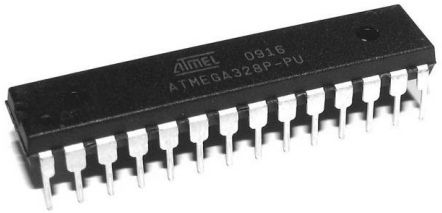
DEPARTMENT OF COMPUTER SCIENCE

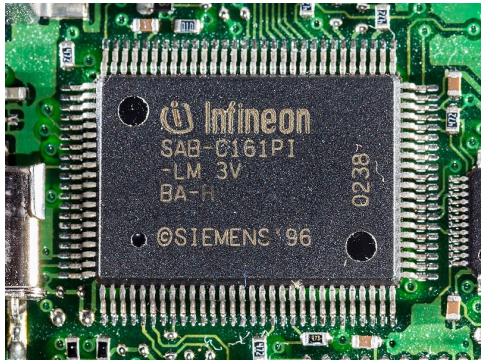
WHAT IS A MICROCONTROLLER?

A MCU (Microcontroller Unit) is a functional computer system-on-a-chip that contains a processor core, memory and programmable input/output peripherals

It has 3 main functions:

- Control, sensing and communication.





From wikimedia commons

DEPARTMENT OF COMPUTER SCIENCE

WHAT IS A MICROCONTROLLER?

Microcontrollers are often defined as being complete computers on a single chip

They have a processor that is similar to a computer CPU.

Like any computer, you have the option of programming the microcontroller using a variety of languages.

DEPARTMENT OF COMPUTER SCIENCE

VERY SMALL!

AVRs have:

- Flash program memory space in kilobytes - 1 KB to 32 KB
- Limited RAM
- 8-bit CPUs without a floating-point math co-processor inside.
  - 0-255 per operation
  - Computer CPUs have FPUs (floating-point processing units) making them able to do decimal math and arithmetic.
  - Simple and dumb – but functional and tiny!

DEPARTMENT OF COMPUTER SCIENCE

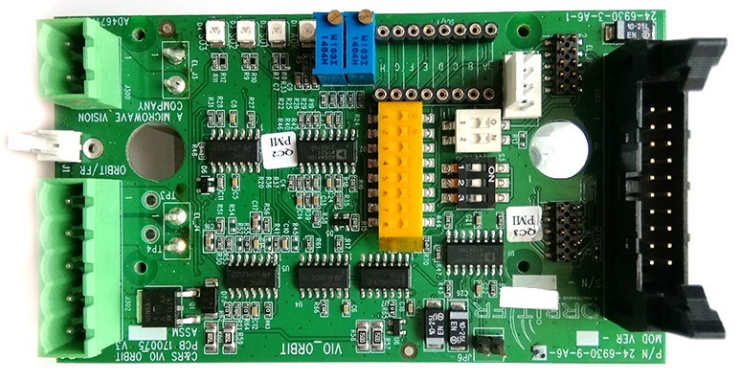
MCUS ARE EVERYWHERE!

About 55% of all CPUs sold in the world are 8-bit microcontrollers.

A typical home in a developed country is likely to have around 40 microcontrollers

- washing machines, microwave ovens, telephones etc.

A standard car has as many as 30 or more microcontrollers.



DEPARTMENT OF COMPUTER SCIENCE

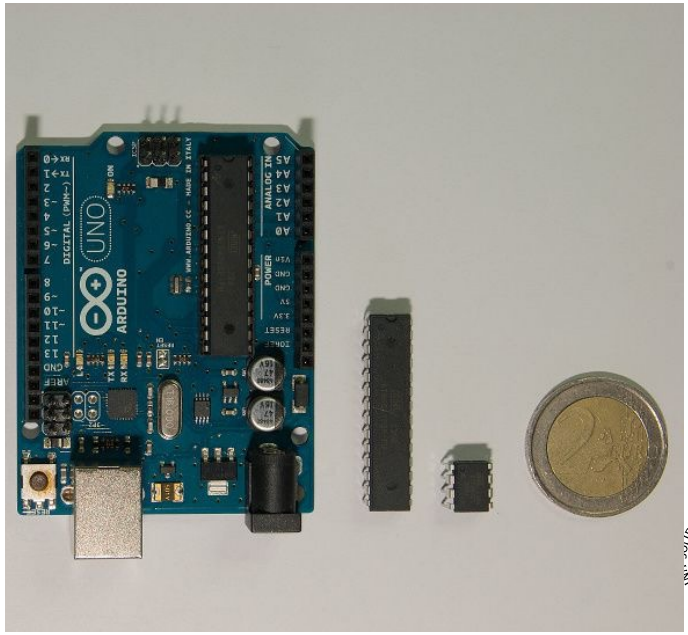
ADVANTAGES AND DISADVANTAGES

Advantages

- size of a circuit can be reduced significantly - one microcontroller can replace several other ICs (integrated circuits)
- allows greater flexibility - can be reprogrammed to change its function

Disadvantages

- they are often more expensive than other ICs
- and all the resource limitations mentioned earlier



DEPARTMENT OF COMPUTER SCIENCE

MICROCONTROLLER TYPES

PIC - Very simple, and widely used.

AVR - Direct competitor of PICs.

MSP - Good for low-power applications.

ARM - Very powerful, and cheap.

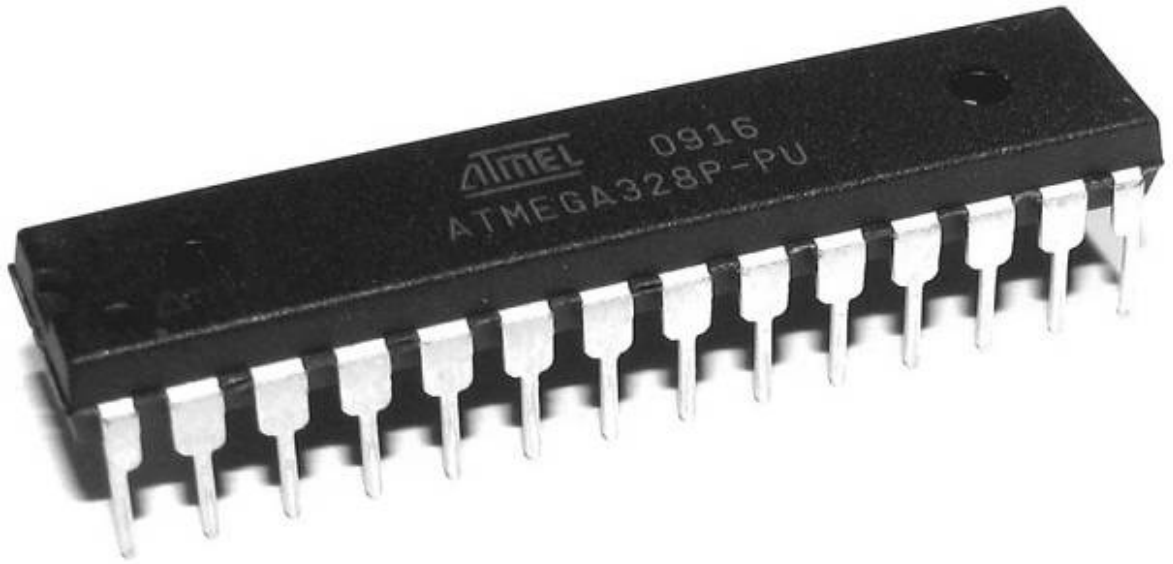
8051 - The '8051 core' was the de facto standard in 8-bit microcontrollers. Developed by Intel in the 1980s.

68HC08/11 - Developed by Motorola. Often lack on-board RAM and flash based memory.

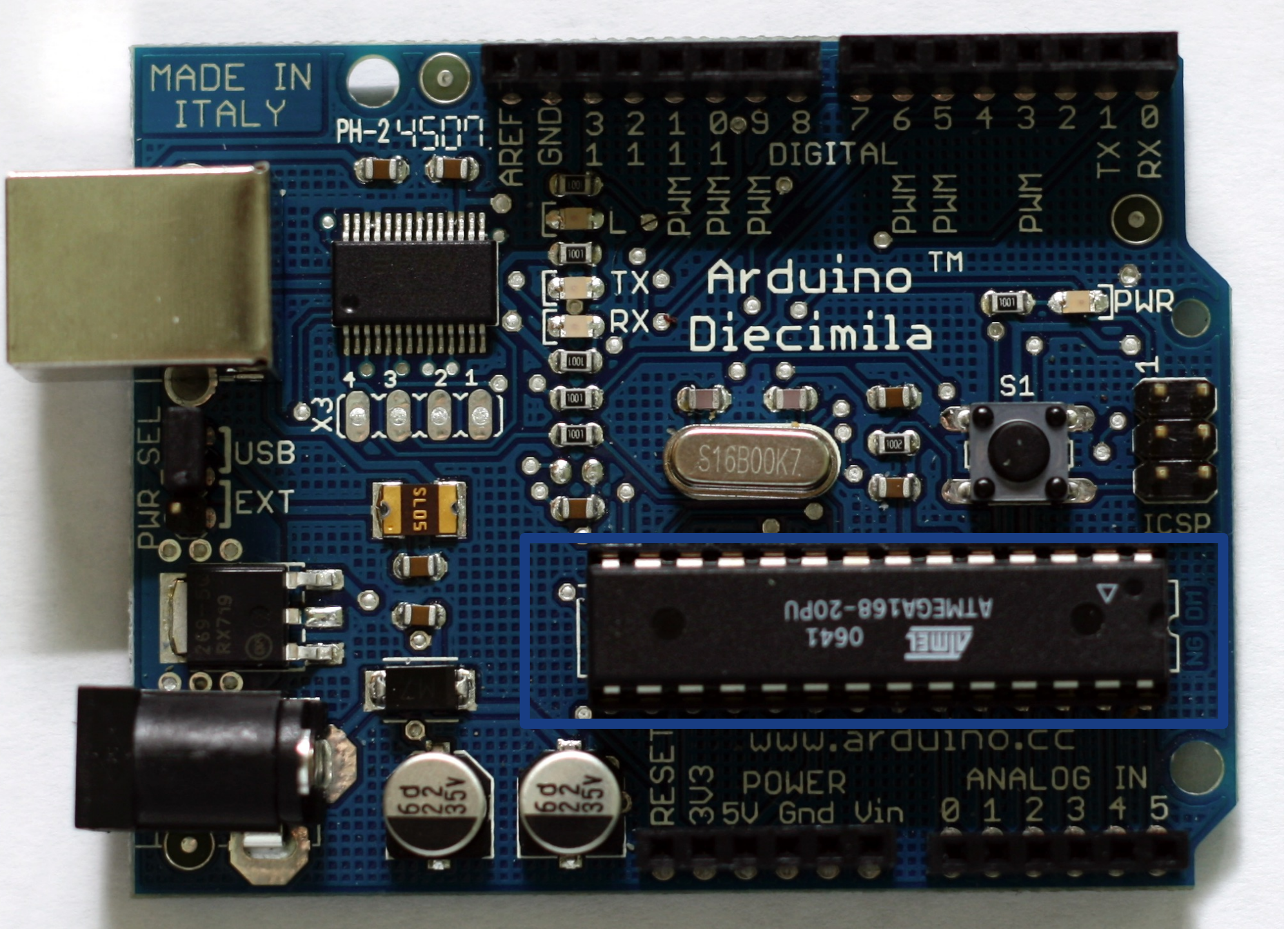
BASIC STAMP, BASICX

DEPARTMENT OF COMPUTER SCIENCE

AVRS



DEPARTMENT OF COMPUTER SCIENCE


MADE IN ITALY
PH-24507
AREF
GND
DIGITAL
TX
RX
Arduino TM
Diecimila
PWR
S1
ICSP
PWR_SEL
USB
EXT
S16B00K7
ATMEGA168-20PU
0641
ATMEL
RESET
3V3
POWER
5V Gnd Vin
ANALOG IN
0 1 2 3 4 5
www.arduino.cc

AVR FAMILY

Small: ATtiny45

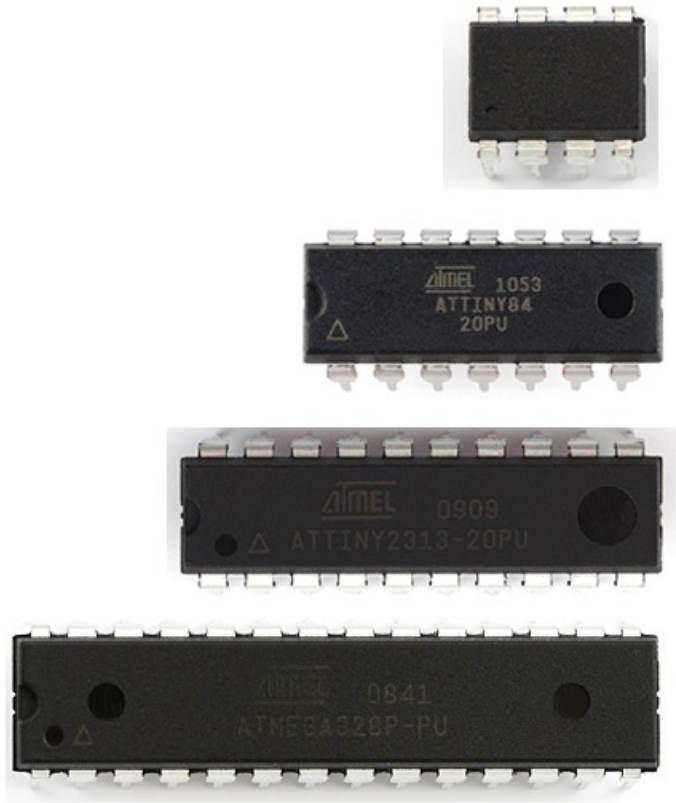
- small and cheap
- five I/O pins.
- high-speed peripheral clock that can run up to 66 MHz, which makes them uniquely great for PWM applications.
- The only differences between the 25, 45, and 85 are the amount of program memory (2 kB, 4 kB, and 8 kB) and the price.

Medium: ATtiny 44

- cheap
- A step up from the 45 when you don't need the high-speed PWM of the Tiny45.
- 11 I/O pins and a 16- bit timer.

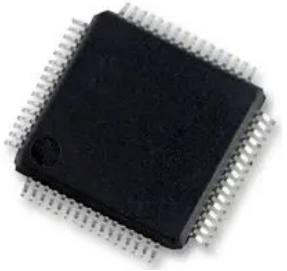
Large: ATmega 328

- >=20 I/O pins,
- hardware USART, three timers, and 32 kB of program memory.



DEPARTMENT OF COMPUTER SCIENCE

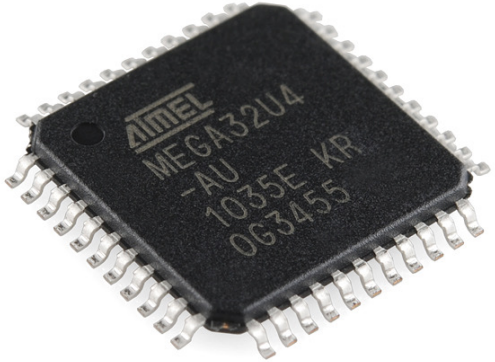
OTHER MICROCONTROLLERS



Muscly: ARM® CortexTM-M4

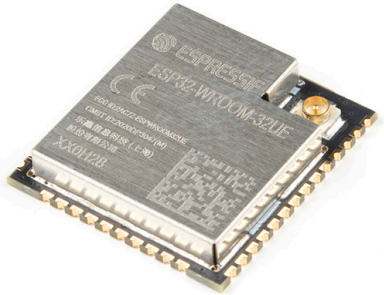
- Featured on the Arduino Nano 33 BLE
- Can run TensorFlow Lite - making low application machine learning a possibility

Specific application: Atmega 32u4



- Featured on the Arduino Pro Micro and Arduino Leonardo
- Can utilize the MOUSE and KEYBOARD libraries
- Makes computer emulation and communication easy

IoT: ESP32



- Integrated WiFi and Bluetooth
- Widely applied in SoCs aimed at Internet of Things
- Easy power consumption control (Deep Sleep)

DEPARTMENT OF COMPUTER SCIENCE

SCHEMATICS AND FEATURES

DEPARTMENT OF COMPUTER SCIENCE

MICROCONTROLLER PINS

Every microcontroller will have

- pins for connecting to power and ground
- pins dedicated to programming the chip
- general input and output (I/O) pins.

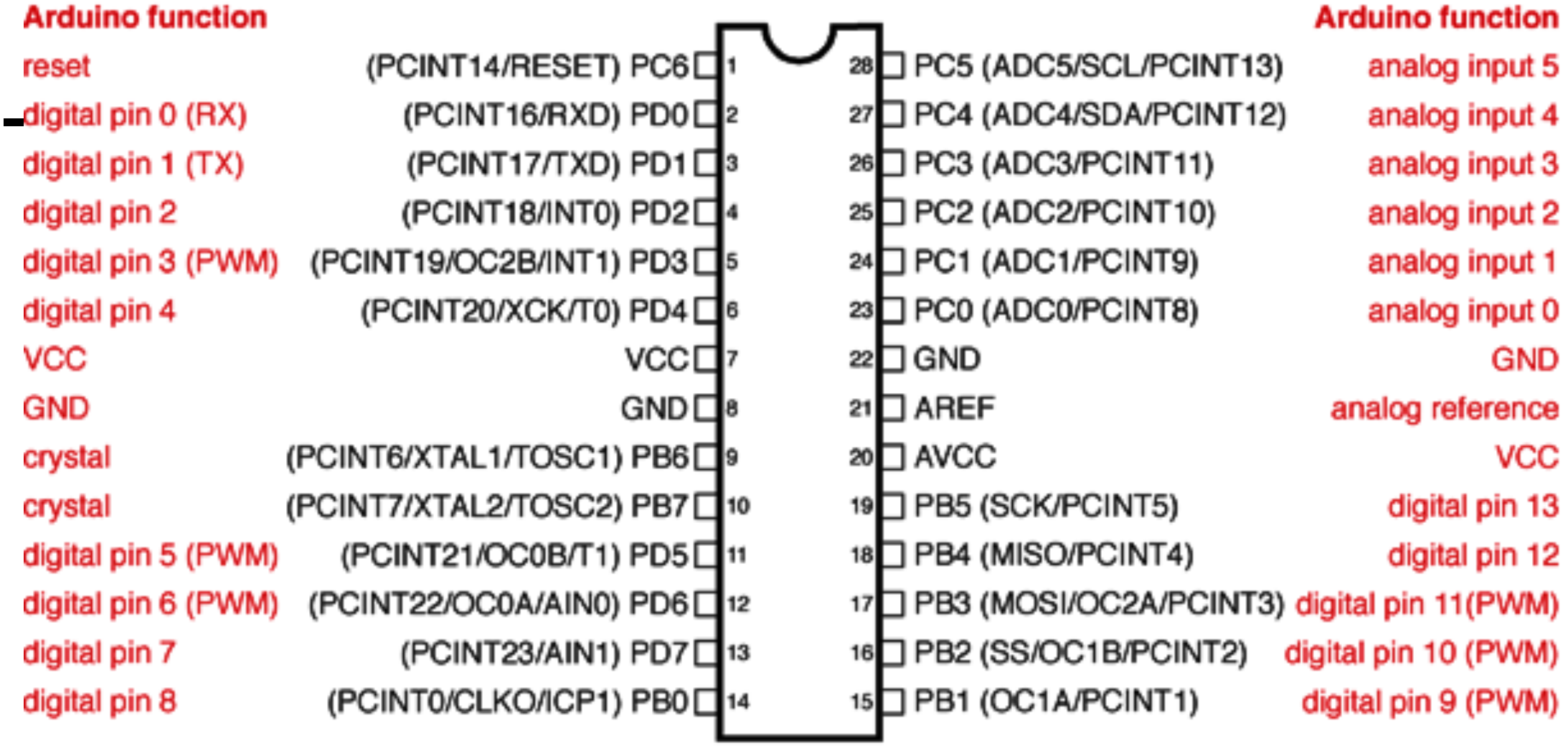
Almost all pins are capable of digital I/O.

Some pins are capable of analog input as well.

On any DIP-packaged microcontroller, the pin numbering starts at the top-left corner

Always look at the datasheet!

DEPARTMENT OF COMPUTER SCIENCE



From Arduino.cc

DEPARTMENT OF COMPUTER SCIENCE

MEMORY

AVR microcontrollers have three different memory types.

Flash: nonvolatile flash memory. Only loses 1 bit per million after 100 years at room temperature.

RAM: for storing temporary variables.

EEPROM: slow to write to, but non-volatile.

DEPARTMENT OF COMPUTER SCIENCE

CLOCKS

Internal RC oscillator:

- master clock source.
- 8 MHz.

There are clocks for the I/O sub- system, the analog-to-digital converter, RAM, and Flash and EEPROM.

ATmega328 – internal 8MHz vs. external 16MHz clock setup

DEPARTMENT OF COMPUTER SCIENCE

COMMUNICATION PROTOCOLS

DEPARTMENT OF COMPUTER SCIENCE

COMMUNICATION PROTOCOLS

I2C (Inter-integrated Circuit) communication

- Single bus with address specific calls using SDA/SCL pins.

SPI (Serial Peripheral Interface)

- MISO/MOSI/SCK/SS pins used for slave/master communication – used to bootload your chips.

UART/USART communication

- RX/TX pins used for serial communication – used when you program the chip and when using Serial Monitor/Plotter. USART uses a separate clock pin.

DEPARTMENT OF COMPUTER SCIENCE

GPIO

General Purpose Input and Output

DEPARTMENT OF COMPUTER SCIENCE

DIGITAL OUTPUT AND INPUT

Output: almost all of the AVR pins can be used as digital outputs.

Input: similarly, they can also be configured as digital inputs.

HIGH = 5V, and LOW = 0V.

DEPARTMENT OF COMPUTER SCIENCE

ANALOG INPUT

Allow you to measure a voltage on an analog pin that lies between the low and high voltages.

For example, in an Arduino, the voltage between 0 and 5V is mapped to a number between 0 and 1023.

- So 0V gives a reading of 0, and 5V a reading of 1023.
- Something in the middle (2.5V) will give a reading of around 511.

DEPARTMENT OF COMPUTER SCIENCE

ANALOG-TO-DIGITAL CONVERTERS (ADC)

Converts an analog signal, such as sound, temperature, or light, into a digital signal.

AVRs have a built-in ADC.

At the heart of the ADC is a DAC—a digital-to-analog converter—and a comparator.

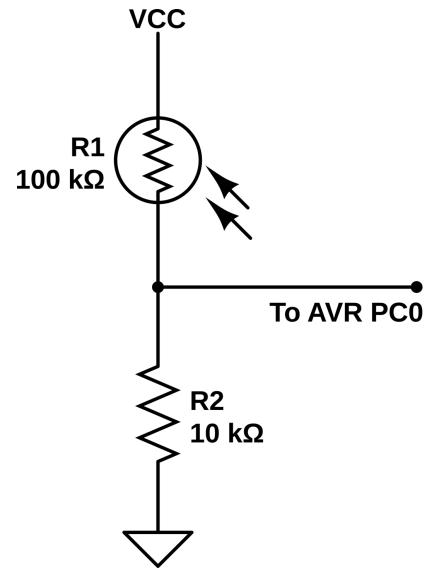
- The DAC generates a voltage that corresponds to a digital value that it takes as input. The comparator is a digital device that outputs a binary high or low value if the input signal coming from one of the ADC pins is higher or lower than the output of the DAC.

Pins PC0 to PC5 are all available for use as ADC inputs

- you can only take readings from one at a time, and you must switch between them using a multiplexer.

DEPARTMENT OF COMPUTER SCIENCE

LIGHT METER – ADC EXAMPLE



LEDs to display the amount of light hitting the sensor.

The sensor portion is basically a *voltage divider*.

When the LDR is less resistive, voltage is higher.

- The LDR gets less resistive when more light shines on it, so there's a direct relationship between the light level and the voltage sent to the AVR

DEPARTMENT OF COMPUTER SCIENCE

PROGRAMMING MICROCONTROLLERS

DEPARTMENT OF COMPUTER SCIENCE

INSTRUCTION SETS

A microcontroller responds to a set of instructions known as its Instruction Set

Do things like:

- Add, Subtract, Multiply, Divide
- Logic
- Jumps, Branches

DEPARTMENT OF COMPUTER SCIENCE

Assembly:

```
      MOV  r0, #0
      MOV  r1, #0
L1:   CMP  r1, #10
      BGE  L2
      ADD  r0,r0,r1
      ADD  r1, r1,#1
      B    L1
L2:   B    L2
```

C/C++:

```
int a = 0;
for(int x=0;x<10;x++){
   a =a+x;
}
```

*colors indicate approximate matching functionalities and operations

DEPARTMENT OF COMPUTER SCIENCE

ENVIRONMENTS

AVR studio

Eclipse

Bascom

Arduino IDE (!!!)

Normally written in C or Assembly Language

DEPARTMENT OF COMPUTER SCIENCE

PROGRAMMING IN ARDUINO

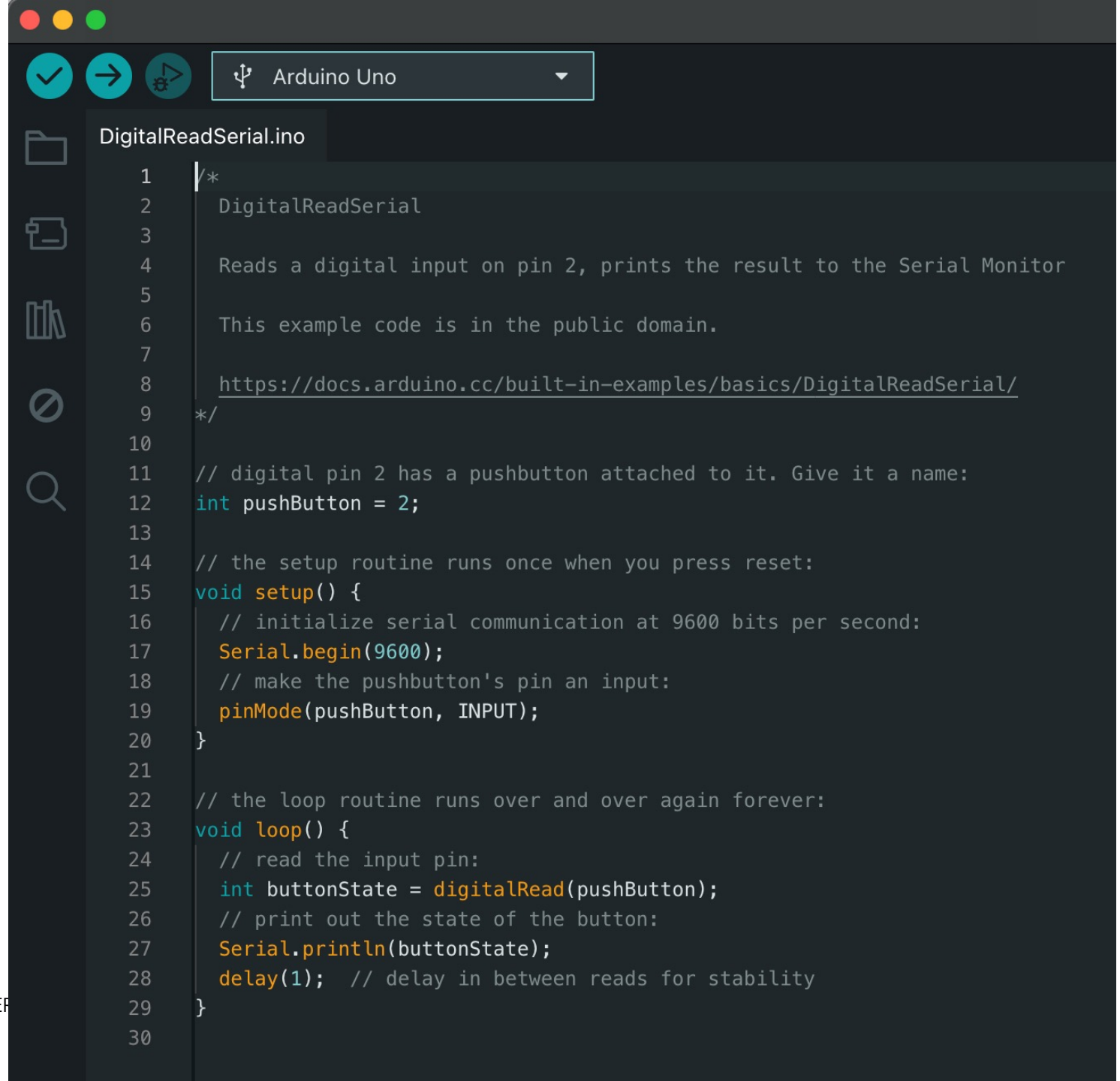
Arduino is ”Arduino C” which is basically C++

Object oriented programming – polymorphism, classes etc.
Never used ☺

Code examples:

- Digital Read Serial
- States + If/else
- Switch case
- For loop
- (Many more, for example under “Examples → Control” in Arduino IDE)

DEPARTMENT OF COMPUTER SCIENCE


Arduino Uno
DigitalReadSerial.ino
/*
DigitalReadSerial
Reads a digital input on pin 2, prints the result to the Serial Monitor
This example code is in the public domain.
https://docs.arduino.cc/built-in-examples/basics/DigitalReadSerial/
*/
// digital pin 2 has a pushbutton attached to it. Give it a name:
int pushButton = 2;
// the setup routine runs once when you press reset:
void setup() {
// initialize serial communication at 9600 bits per second:
Serial.begin(9600);
// make the pushbutton's pin an input:
pinMode(pushButton, INPUT);
}
// the loop routine runs over and over again forever:
void loop() {
// read the input pin:
int buttonState = digitalRead(pushButton);
// print out the state of the button:
Serial.println(buttonState);
delay(1); // delay in between reads for stability
}

```ino
// this constant won't change:
const int buttonPin = 2;  // the pin that the pushbutton is attached to
const int ledPin = 13;    // the pin that the LED is attached to

// Variables will change:
int buttonPushCounter = 0;  // counter for the number of button presses
int buttonState = 0;        // current state of the button
int lastButtonState = 0;    // previous state of the button

void setup() {
  // initialize the button pin as a input:
  pinMode(buttonPin, INPUT);
  // initialize the LED as an output:
  pinMode(ledPin, OUTPUT);
  // initialize serial communication:
  Serial.begin(9600);
}


void loop() {
  // read the pushbutton input pin:
  buttonState = digitalRead(buttonPin);

  // compare the buttonState to its previous state
  if (buttonState != lastButtonState) {
    // if the state has changed, increment the counter
    if (buttonState == HIGH) {
      // if the current state is HIGH then the button went from off to on:
      buttonPushCounter++;
      Serial.println("on");
      Serial.print("number of button pushes: ");
      Serial.println(buttonPushCounter);
    } else {
      // if the current state is LOW then the button went from on to off:
      Serial.println("off");
    }
    // Delay a little bit to avoid bouncing
    delay(50);
  }
  // save the current state as the last state, for next time through the loop
  lastButtonState = buttonState;


  // turns on the LED every four button pushes by checking the modulo of the
  // button push counter. the modulo function gives you the remainder of the
  // division of two numbers:
  if (buttonPushCounter % 4 == 0) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```


DEPARTMENT OF COMPUTER SCIENCE

```ino
// these constants won't change. They are the lowest and highest readings you
// get from your sensor:
const int sensorMin = 0;    // sensor minimum, discovered through experiment
const int sensorMax = 600;  // sensor maximum, discovered through experiment

void setup() {
  // initialize serial communication:
  Serial.begin(9600);
}

void loop() {
  // read the sensor:
  int sensorReading = analogRead(A0);
  // map the sensor range to a range of four options:
  int range = map(sensorReading, sensorMin, sensorMax, 0, 3);

  // do something different depending on the range value:
  switch (range) {
    case 0:  // your hand is on the sensor
      Serial.println("dark");
      break;
    case 1:  // your hand is close to the sensor
      Serial.println("dim");
      break;
    case 2:  // your hand is a few inches from the sensor
      Serial.println("medium");
      break;
    case 3:  // your hand is nowhere near the sensor
      Serial.println("bright");
      break;
  }
  delay(1);  // delay in between reads for stability
}
```

```ino
int timer = 100;  // The higher the number, the slower the timing.

void setup() {
  // use a for loop to initialize each pin as an output:
  for (int thisPin = 2; thisPin < 8; thisPin++) {
    pinMode(thisPin, OUTPUT);
  }
}

void loop() {
  // loop from the lowest pin to the highest:
  for (int thisPin = 2; thisPin < 8; thisPin++) {
    // turn the pin on:
    digitalWrite(thisPin, HIGH);
    delay(timer);
    // turn the pin off:
    digitalWrite(thisPin, LOW);
  }

  // loop from the highest pin to the lowest:
  for (int thisPin = 7; thisPin >= 2; thisPin--) {
    // turn the pin on:
    digitalWrite(thisPin, HIGH);
    delay(timer);
    // turn the pin off:
    digitalWrite(thisPin, LOW);
  }
}
```

INTERACTION DESIGN WITH MICROCONTROLLERS

DEPARTMENT OF COMPUTER SCIENCE

INTERACTION DESIGN

Embedded interactive systems are getting increasingly widespread.

- Smart objects, IoT, mobile and wearable computing

Embedded microcontrollers make the design and development faster and easier.

These are essential skills for developing the next generation of human-computer interfaces!

DEPARTMENT OF COMPUTER SCIENCE

INTERACTION DESIGN

It is crucial to remember that you're working with:

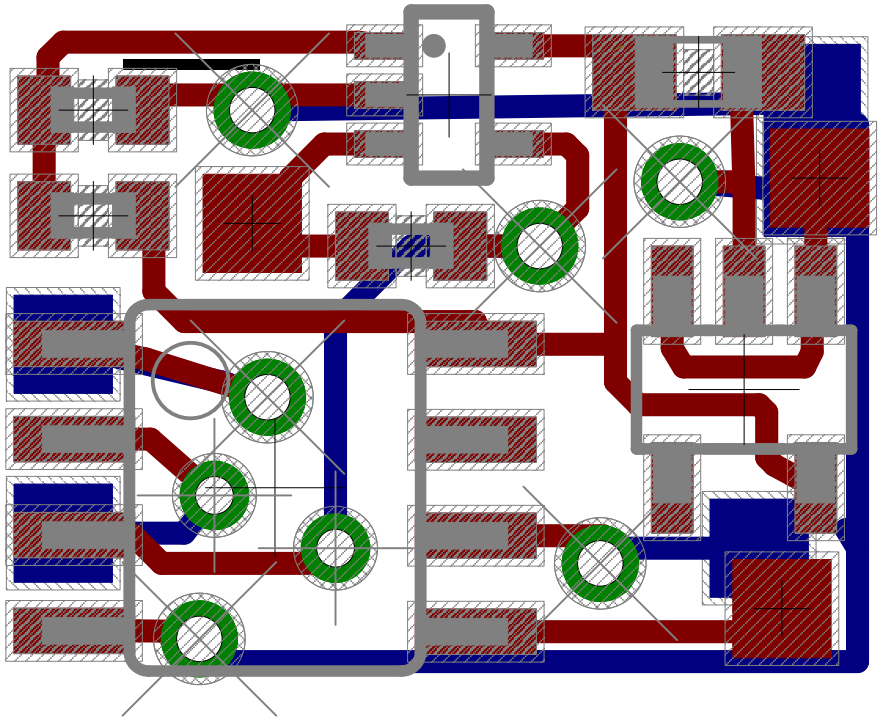
- Limited memory
- Limited pins

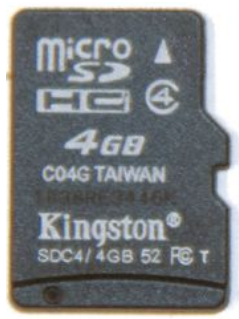
For complex interaction, you often need to offload the computation somewhere else

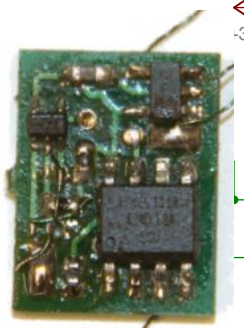
DEPARTMENT OF COMPUTER SCIENCE

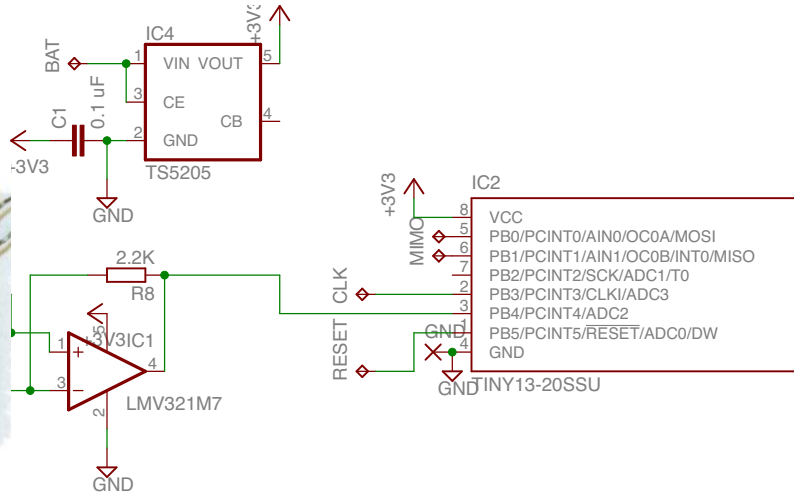
FORCEPHONE

DEPARTMENT OF COMPUTER SCIENCE




micro SD HC
4GB
C04G TAIWAN
Kingston
SDC4/4GB 52




IC4
VIN VOUT
CE
CB
GND
TS5205
BAT
C1
0.1 uF
-3V3
GND
2.2K
R8
3V3IC1
LMV321M7
GND
+3V3
IC2
MIMO
CLK
RESET
VCC
PB0/PCINT0/AIN0/OC0A/MOSI
PB1/PCINT1/AIN1/OC0B/INT0/MISO
PB2/PCINT2/SCK/ADC1/T0
PB3/PCINT3/CLKI/ADC3
PB4/PCINT4/ADC2
PB5/PCINT5/RESET/ADC0/DW
GND
GND
TINY13-20SSU

SUMMARY

Microcontrollers

- Uses
- Types

Schematics and Features

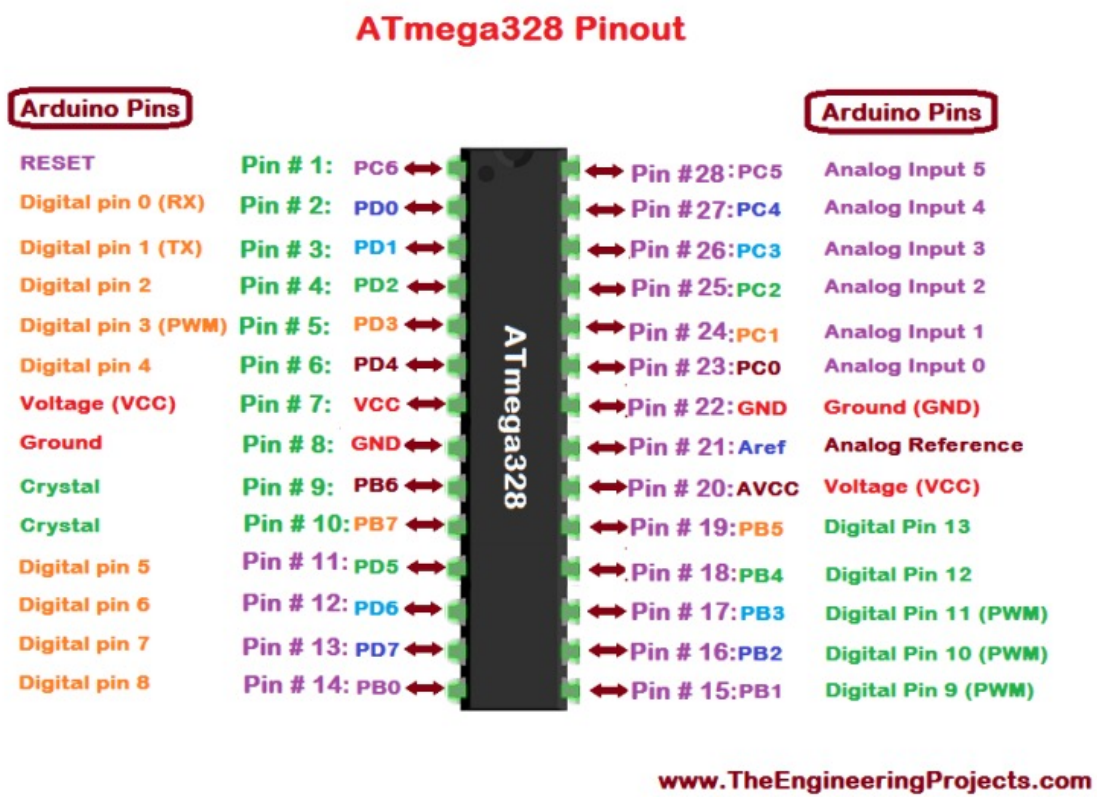
ADC

Interaction Design with Microcontrollers

DEPARTMENT OF COMPUTER SCIENCE

PRACTICAL EXERCISE

ATMEGA 328PU

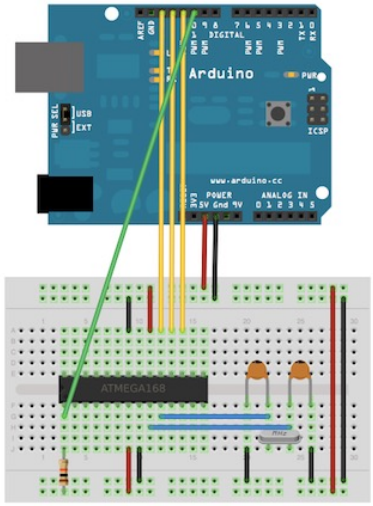


TASK 1 – BURN BOOTLOADER

Update to latest Arduino IDE

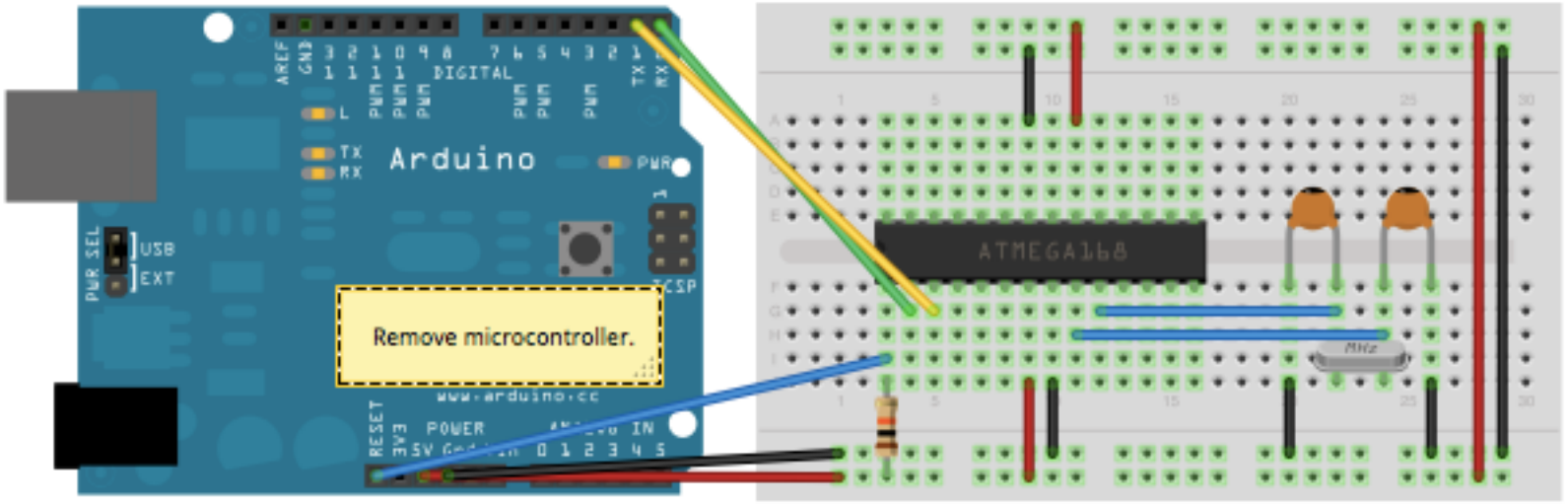
Follow the instructions here or on the more detailed walkthrough BS→ Practical Exercises.

- https://docs.arduino.cc/built-in-examples/arduino-isp/ArduinoToBreadboard/



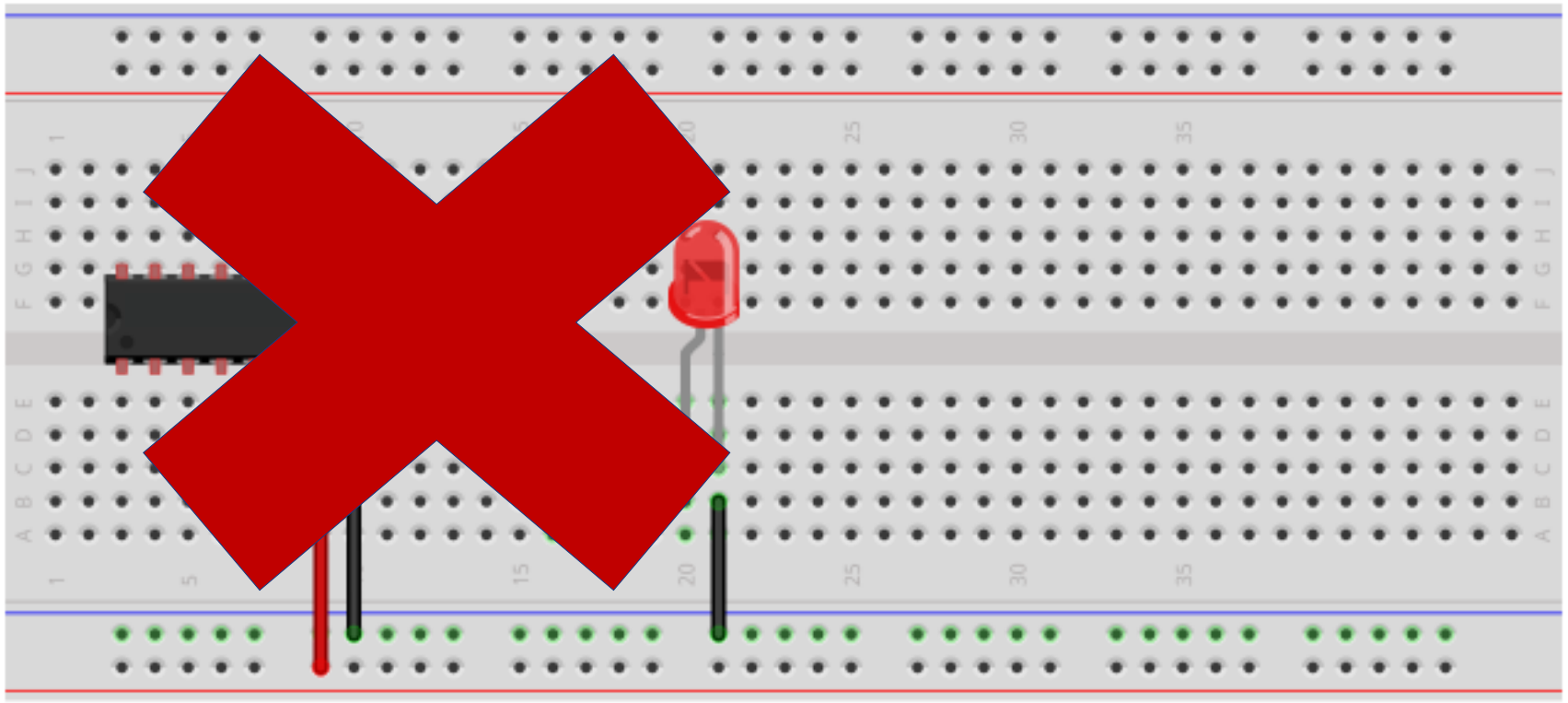
TASK 2 - UPLOAD

- https://docs.arduino.cc/built-in-examples/arduino-isp/ArduinoToBreadboard/



TASK 3 – BLINK LED

https://www.arduino.cc/en/Tutorial/Blink



DEBUGGING

Working setups for each step.

Can be used for variable control – but do it controlled and keep the functional setups intact.

Double, triple and quadruple-check your wiring. It is often the problem.

Problematic components to test:
- 10k resistor (important)
- Crystal (16mHz) + caps (18-22pf, try a few different ones)
- Arduino (if borrowed from Labtools, some of them are wonky)

MOST COMMON ATMEGA AND ARDUINO ISSUES

Make sure to keep the original chip from the Arduino. A newly bootloaded chip will "understand" Arduino sketches, but can't replace the original chip (it's not an UNO).

Make sure to upload the ArduinoISP sketch to the regular UNO first

avrdude: stk500_recv(): programmer is not responding

avrdude: stk500_getsync() attempt <n> of 10: not in sync

Often the result of incorrect wiring.

Test if the "Nano --> 328p" works instead of "Duemilanove"

DEPARTMENT OF COMPUTER SCIENCE

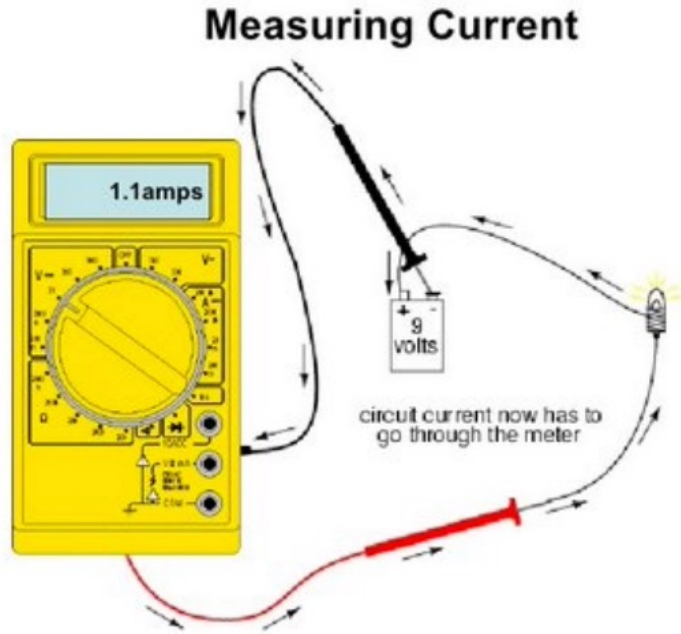
HOW TO USE A MULTIMETER

https://learn.sparkfun.com/tutorials/how-to-use-a-multimeter

Measure voltages in parallel (plus to plus, minus to minus)

Measure amps in series – remember that a load is necessary (see picture)

Measuring ohms one probe on each leg (on resistors) or non-wiper leg on pots.



DEPARTMENT OF COMPUTER SCIENCE

DEPARTMENT OF COMPUTE