


ROCO222: Intro to sensors and actuators

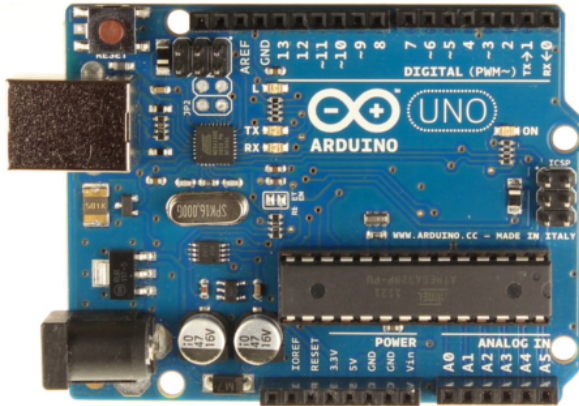
Lecture 4

Arduino microcontroller

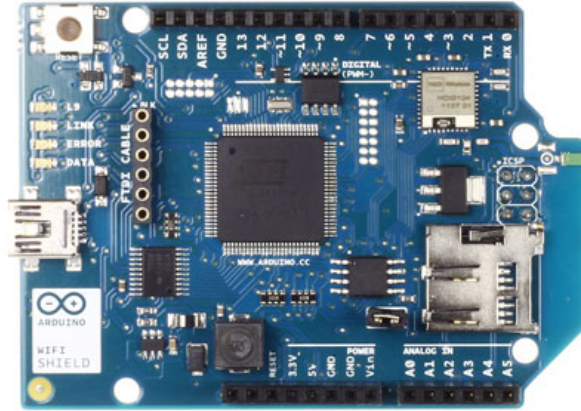
The Arduino microcontroller

- Intro to Arduinos
 - Developing programs
 - Inputs and Outputs
 - Running control loops
 - Using PWM to control motors
- 
- The image shows an Arduino Uno microcontroller board, which is a blue printed circuit board (PCB) populated with various electronic components. Key features include a USB Type-B port on the left, a DC power jack at the bottom left, and two rows of pin headers. The top header is labeled 'DIGITAL (PWM~)' and the bottom header is labeled 'ANALOG IN'. The board is marked with the Arduino logo and the word 'UNO'. Various integrated circuits, including the ATmega328P microcontroller, and passive components like resistors and capacitors are visible on the surface.
- Arduino is an open-source electronics platform based on easy-to-use hardware and software
 - It's intended for anyone making interactive projects

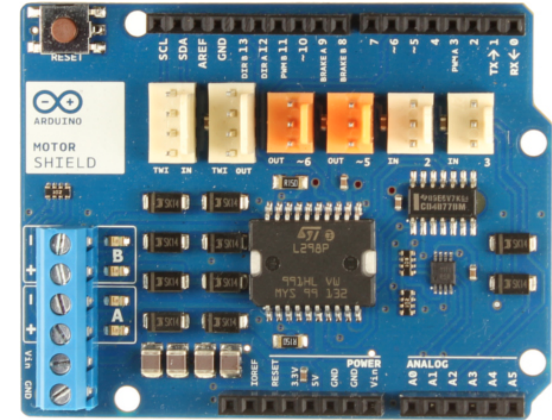
The Arduino microcontroller system



Controller



Wi-Fi interface



Motor controller

<http://arduino.cc/en/>



Ultrasonic sensors



Robot car

“Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists and anyone interested in creating interactive objects or environments”

Arduino boards

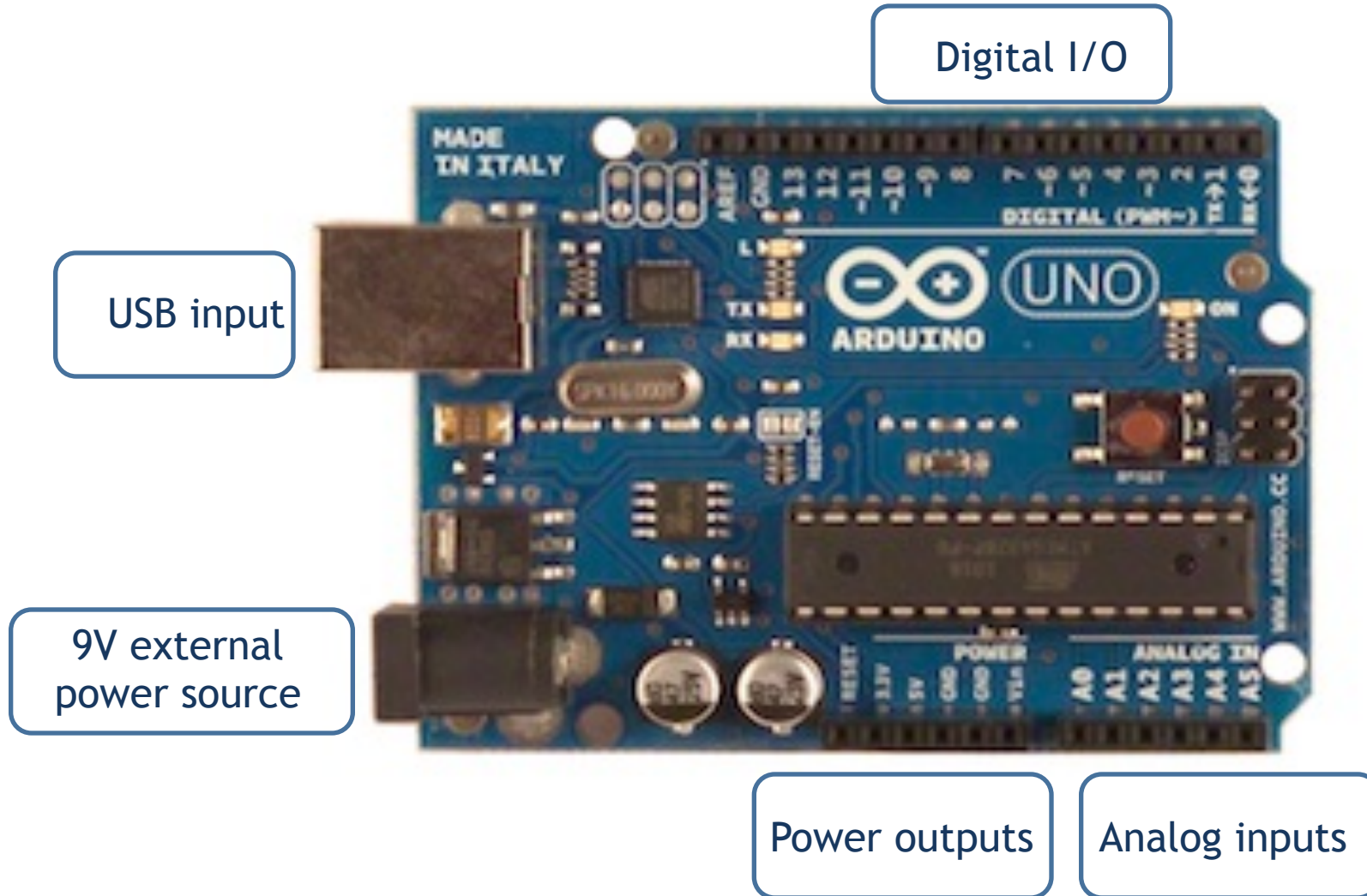
Name	Processor	Operating Voltage/Input Voltage	CPU Speed	Analog In/Out	Digital IO/PWM	EEPROM [KB]	SRAM [KB]	Flash [KB]	USB	UART
Uno	ATmega328	5 V/7-12 V	16 Mhz	6/0	14/6	1	2	32	Regular	1
Due	AT91SAM3X8E	3.3 V/7-12 V	84 Mhz	12/2	54/12	-	96	512	2 Micro	4
Leonardo	ATmega32u4	5 V/7-12 V	16 Mhz	12/0	20/7	1	2.5	32	Micro	1
Mega 2560	ATmega2560	5 V/7-12 V	16 Mhz	16/0	54/15	4	8	256	Regular	4
Mega ADK	ATmega2560	5 V/7-12 V	16 Mhz	16/0	54/15	4	8	256	Regular	4

Arduino Uno

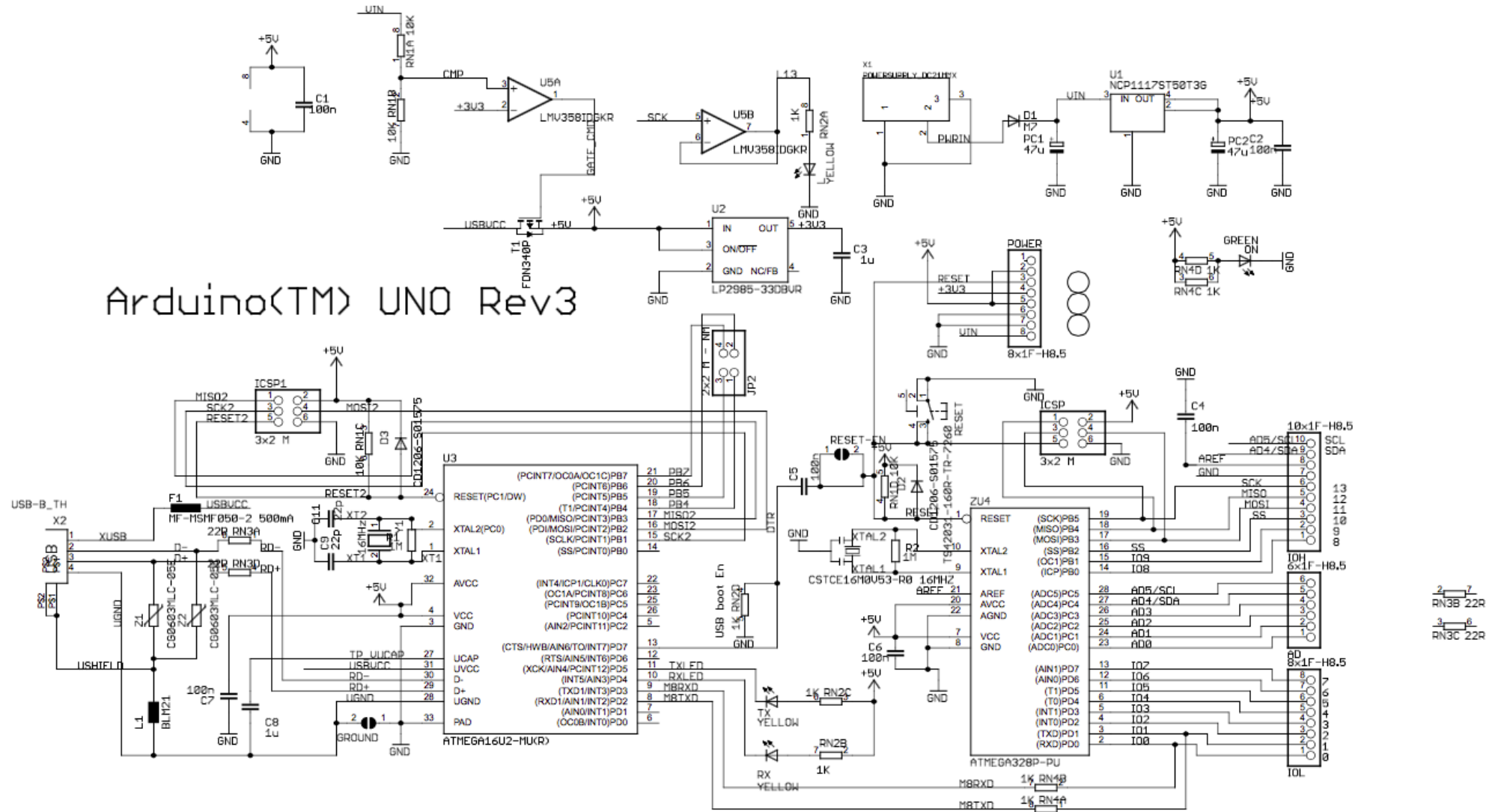
- Relatively robust board
- Microcontroller board based on the ATmega328
- 14 digital input/output pins
- 6 output can be used as PWM outputs
- 6 analog inputs
- 16 MHz ceramic resonator
- USB connection
- Power jack
- ICSP header
- Reset button



Arduino Uno



Arduino Uno schematic



Arduino Uno summary

• Microcontroller	ATmega328
• Operating Voltage	5V
• Input Voltage (recommended)	7-12V
• Input Voltage (limits)	6-20V
• Digital I/O Pins (of which 6 provide PWM output)	14
• Analog Input Pins	6
• DC Current per I/O Pin	40 mA
• DC Current for 3.3V Pin	50 mA
• Operate at 5 volts	
• Pins have internal pull-up resistor (disconnected by default) of 20-50 kOhms	
• Flash Memory (of which 0.5 KB used by bootloader)	32 KB (ATmega328)
• SRAM	2 KB (ATmega328)
• EEPROM	1 KB (ATmega328)
• Clock Speed	16 MHz

Powering Arduino Uno

- Via the USB connection
- External power supply
- The power source is selected automatically.
- External (non-USB) 2.1mm center-positive plug into the board's power jack
- Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector
- Recommended range is 7 to 12 volts

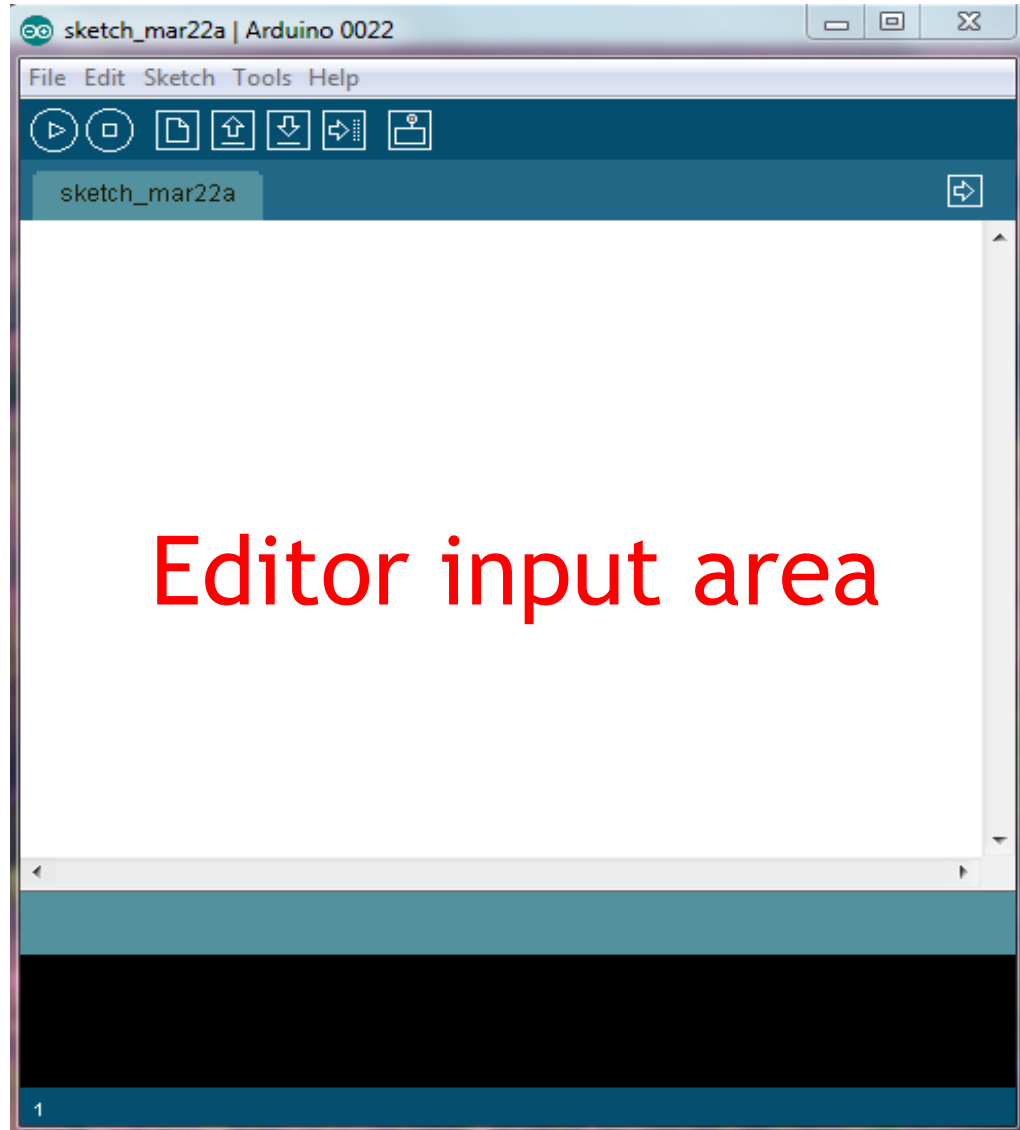
Some pins have specialized functions

- Serial: 0 (RX) and 1 (TX)
- Used to receive (RX) and transmit (TX) TTL serial data
- External Interrupts: 2 and 3
- Can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value
- PWM: 3, 5, 6, 9, 10, and 11
- Provide 8-bit PWM output with the `analogWrite()` function

Some pins have specialized functions

- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK)
- Support SPI communication using SPI library
- LED: 13. There is a built-in LED connected to digital pin 13
- The Uno has 6 analog inputs, labeled A0 through A5
- TWI: A4 or SDA pin and A5 or SCL pin.
- Support TWI communication using the Wire library.

How to start coding: Software setup



- Open the Arduino main window shown on the left
- Download is available at www.arduino.cc
- Use a USB A to B cable to plug your computer into the Arduino
- Setup the port connection
- Setup the board type

Programming the Arduino

- Arduino's programming language is a version of C
- Most C commands work in the sketch editor
- Four basic components of Arduino program
 - Initialization
 - Setup
 - Loop
 - User defined functions
- Arduino Uno is designed to allow it to be reset by software running on a connected computer
- One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor

Programming the Arduino: Initialization

- Include any extra libraries you are using
- All variables need to be initialized
- `A=4;` will not work `int a= 4;` will
- Define variables you will assign later
 - `Int a,b,c;`
- Remember data structures
 - `Int, double, float, char, string`

Programming the Arduino: setup

- Initialization for the Arduino board
- Usually begin Serial communication
- Usually assign digital pins to input or output
- For example:

```
void setup()  
{  
  pinMode(13, OUTPUT);  
  pinMode(12, INPUT);  
  Serial.begin(9600);  
}
```

Void means setup will not output anything

pinMode defines the state of digital pins to OUTPUT or INPUT.

Serial.begin(9600) opens serial communication at 9600 baud

Programming the Arduino: loop

- Loop is the main portion of the Arduino code
- Loop is what the microcontroller will do forever
- This is very much like a never ending for loop

```
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  
}
```

Programming the Arduino: user defined functions

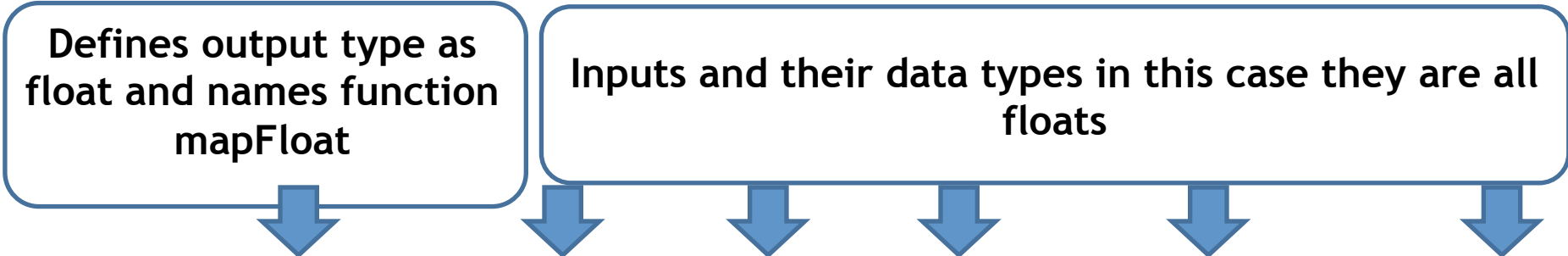
- These functions can be called in the loop and are usually responses to conditions that are met in the loop.
- These functions must be defined with data structure and output type
- Here is a basic map function for floats
- The user inputs a float x and the function maps the number from an old data range to a new data range

```
float mapFloat(float x, float in_min, float in_max, float out_min, float out_max)
{
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

Programming the Arduino: user defined functions

Defines output type as float and names function mapFloat

Inputs and their data types in this case they are all floats



```
• float mapFloat(float x, float in_min, float in_max, float out_min, float out_max)
• {
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
• }
```

- Return is what the function output which is defined above as a float
- The curly brackets { and } define what is inside the function these separate blocks of code and are very important
- You can indent the inside of the code it is up to you and what fits your style

Programming the Arduino: digital I/O

- Each of the 14 digital pins on the Uno can be used as an input or output, using supplied functions
- `pinMode(pin,mode)`
 - Set a digital pin as INPUT or OUTPUT
- `digitalWrite(pin,state)`
 - Write to a digital pin HIGH 5v or LOW 0v or ground
- `digitalRead(pin)`
 - Reads a digital pin and returns high or low (1 or 0)

Programming the Arduino: analogue I/O

- `analogReference(type)`
 - Default is 5 V signal External is voltage applied to VREF
- `analogRead(port)`
 - Reads data coming into analog port, 1024 values or 10 bit data
- <http://arduino.cc/en/Reference/AnalogReference>

Programming the Arduino: serial communications

- The ATmega328 provides UART TTL serial communication
- Available on digital pins 0 (RX) and 1 (TX)
- RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer
- Arduino software includes a serial monitor
- Allows simple text to be sent /received

Programming the Arduino: serial library

- `Serial.begin(baudrate)`
 - Opens Serial Port at baudrate
- `Serial.available()`
 - Returns the number of bytes available to read
- `Serial.read()`
 - Returns the first byte of incoming serial data available (or -1 if no data is available)
- `Serial.print(data,format)`
 - Prints data to the serial port as readable ASCII text
- `Serial.println(data,format)`
- Prints data to the serial port as readable ASCII text then ends the line
- <http://arduino.cc/en/Reference/Serial>

Programming the Arduino: simple control

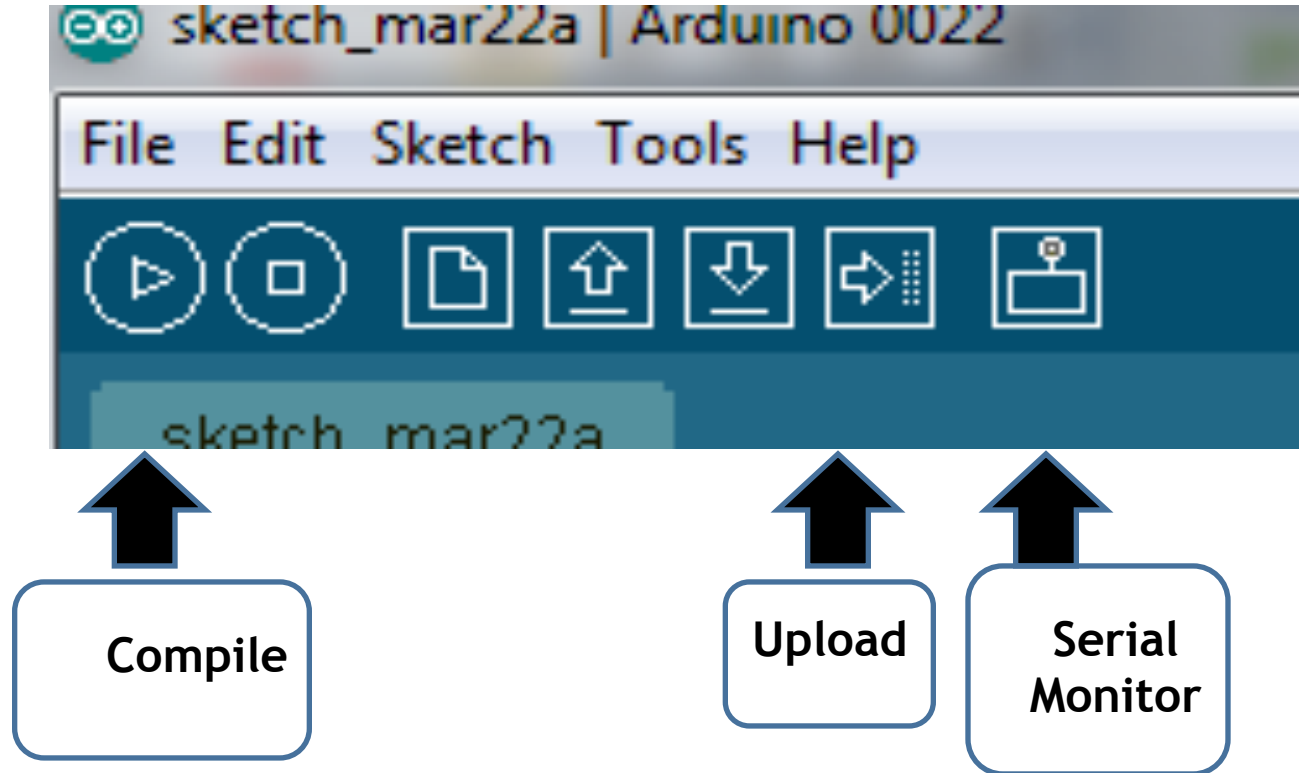
```
If(variable comparison operator condition)
{
    do something
}
```

Computer checks to see if the condition is met for the variable and responds accordingly

```
for(initialization; condition; increment)
{
    do something
}
```

Computer does something for a certain number of time for example read 500 lines of data very quickly

Compiling and uploading code



Under tools make sure Board is the board you are using and Serial port is checked and is the correct port

ROCO222: Intro to sensors and actuators

Lecture 4

Example Arduino programs

Arduino blink led code

```
/* Blink: Turns on an LED on for one second, then off for one second, repeatedly.
```

```
This example code is in the public domain. */
```

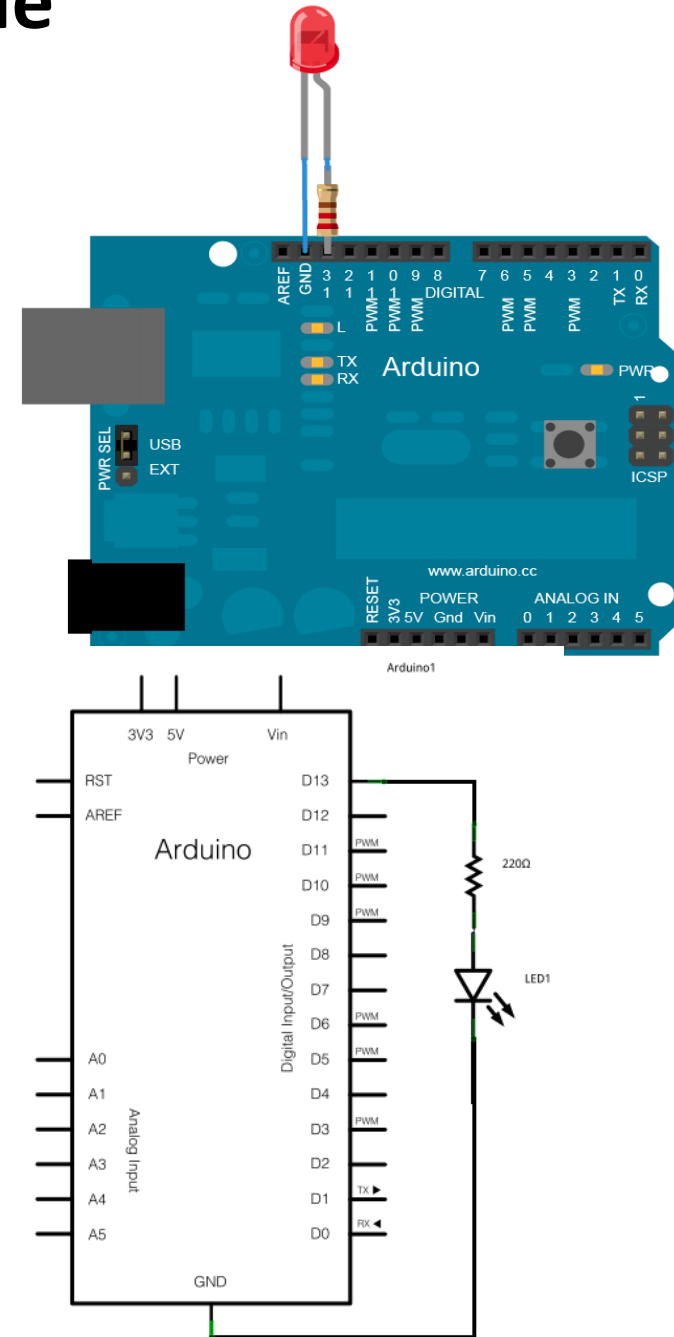
```
// Pin 13 has an LED connected on  
// most Arduino boards.  
// give it a name:  
int led = 13;
```

```
// the setup routine runs once when you press reset:
```

```
void setup() {  
    // initialize the digital pin as an output.  
    pinMode(led, OUTPUT);  
}
```

```
// the loop routine runs over and over again forever:
```

```
void loop() {  
    digitalWrite(led, HIGH);    // turn the LED on (HIGH is the  
                                // voltage level)  
    delay(1000);                // wait for a second  
    digitalWrite(led, LOW);     // turn the LED off by making  
                                // the voltage LOW  
    delay(1000);                // wait for a second  
}
```



Arduino blink LED code

```
/* Blink: Turns on an LED on for one second, then off for one second, repeatedly.
```

```
This example code is in the public domain. */
```

```
// Pin 13 has an LED connected on  
// most Arduino boards.  
// give it a name:  
int led = 13;
```

```
// the setup routine runs once when you press reset:
```

```
void setup() {  
    // initialize the digital pin as an output.  
    pinMode(led, OUTPUT);  
}
```

```
// the loop routine runs over and over again forever:
```

```
void loop() {  
    digitalWrite(led, HIGH);    // turn the LED on (HIGH is the  
                                // voltage level)  
    delay(1000);                // wait for a second  
    digitalWrite(led, LOW);     // turn the LED off by making  
                                // the voltage LOW  
    delay(1000);                // wait for a second  
}
```

Blink explained

There are no variables assigned in this code so the first part is the setup

We are not using any Serial communication so we only need to initialize the pinMode

Writing the pin state to High will turn the LED on the board on, writing it to Low will turn the LED off
The delay is in ms which causes the blink