MITMECHE

Introduction to Embedded System Programing

Workshop TiM THE BEAVER

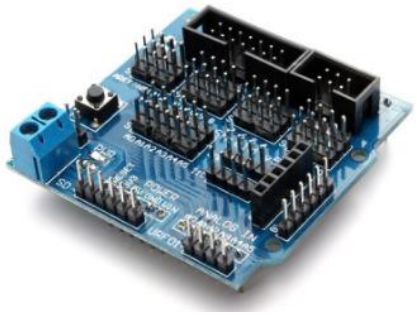
Department of Mechanical Engineering
Massachusetts Institute of Technology

Outline

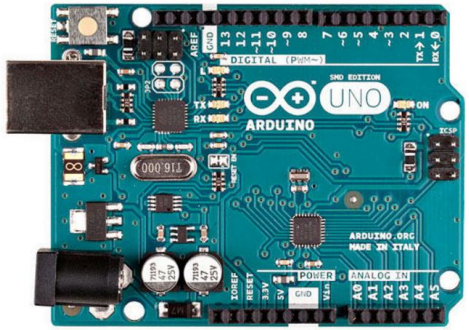
- Embedded System Overview
  - Types of embedded hardware
  - Controller options
- Introduction to Arduino
  - Arduino hardware resources
  - Installation of Arduino IDE
- Programing with Arduino
  - Introduction to C/C++ language
  - Learning Arduino by example
- Introduction to Python
  - Python overview
  - Python programming

Embedded Hardware

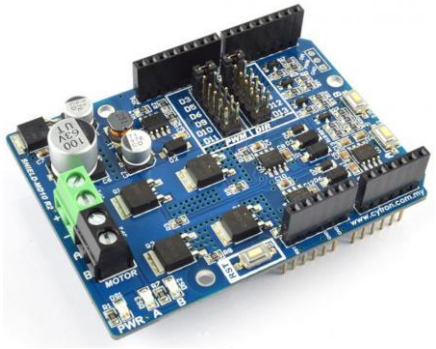
- Embedded system is electrical components in mechatronic systems
- DAQs: Sensor data acquisition boards
- Actuator driver: controller to actuator power amplifier
- Micro-controller: programmable mini-computer



Sensor Shield



Arduino Uno

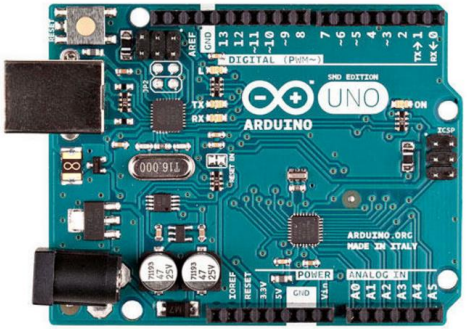


Motor Driver

Controller Options

- Embedded system resources varies greatly depending on application
- ARM Cortex-M7 board is used widely in industry
- Arduino Uno is commonly used for its convenience
- MyRIO contains Field Programmable Gate Array for faster calculation

ARM Cortex-M7 Board



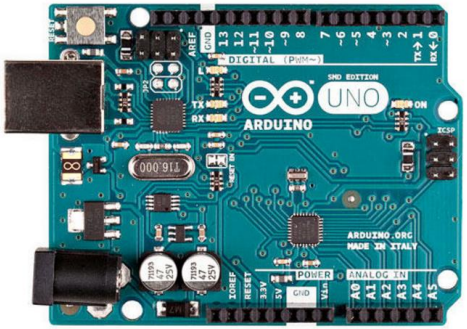
Arduino Uno

NI MyRIO

Introduction to Arduino

- Arduino is open hardware eco-system starting from 2005
- Widely used in projects for students sand hobbyists
- Strong online community
- Different size and resources available

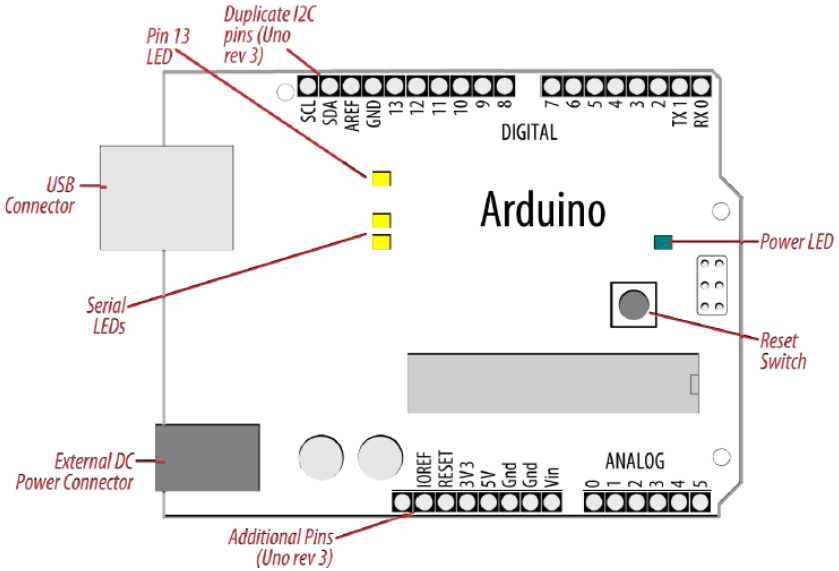
Arduino Mega



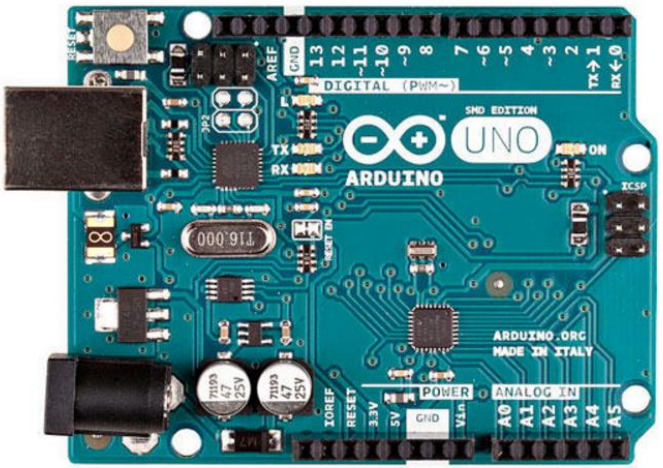
Arduino Uno

Arduino Nano

Arduino Uno Hardware Resources



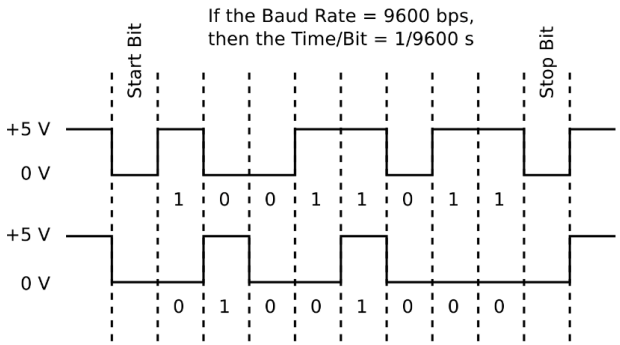
Arduino Uno Resource



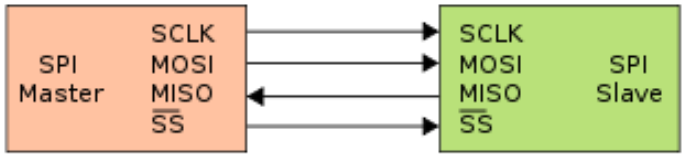
Arduino Uno

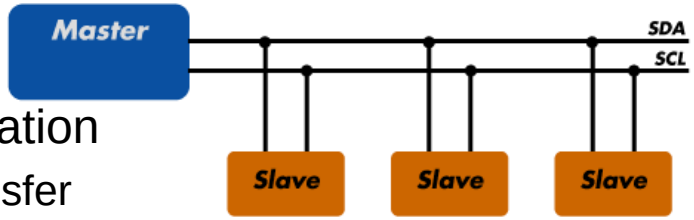
Communication Protocol

- Serial Communication
  - TxD for sending data
  - RxD for receiving data
  - Point to point data transfer
  - Baud rate for data rate
  - Data encoded in packets
- Serial Peripheral Interface Bus
  - SCLK : Serial Clock (output from master)
  - MOSI : Master Output, Slave Input
  - MISO : Master Input, Slave Output
  - SS : Slave Select (output from master).
- Inter-Integrated Circuit (I2C) Communication
  - Use data line and clock line for data transfer

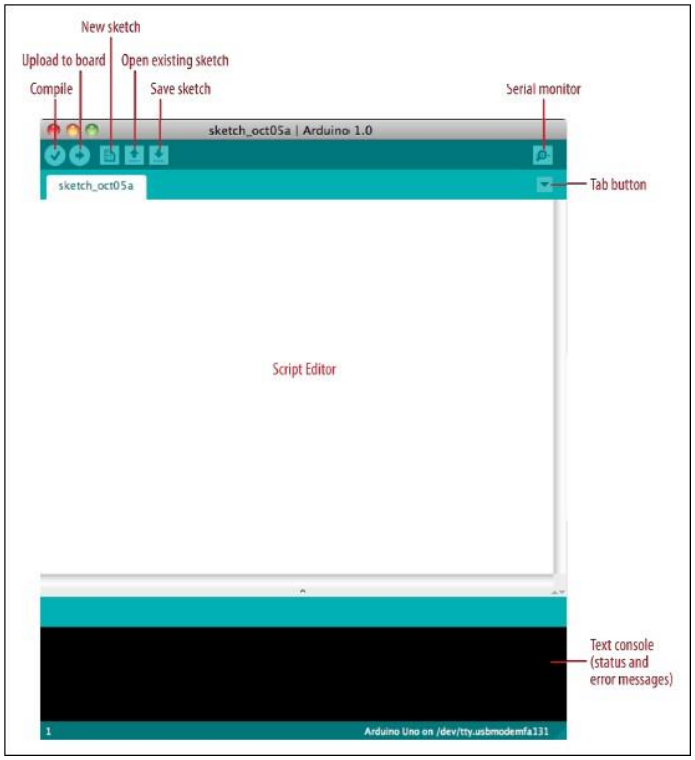


Packet





Arduino IDE


New sketch
Upload to board
Open existing sketch
Compile
Save sketch
Serial monitor
sketch_oct05a | Arduino 1.0
sketch_oct05a
Tab button
Script Editor
Text console
(status and
error messages)
1
Arduino Uno on /dev/tty.usbmodemfa131

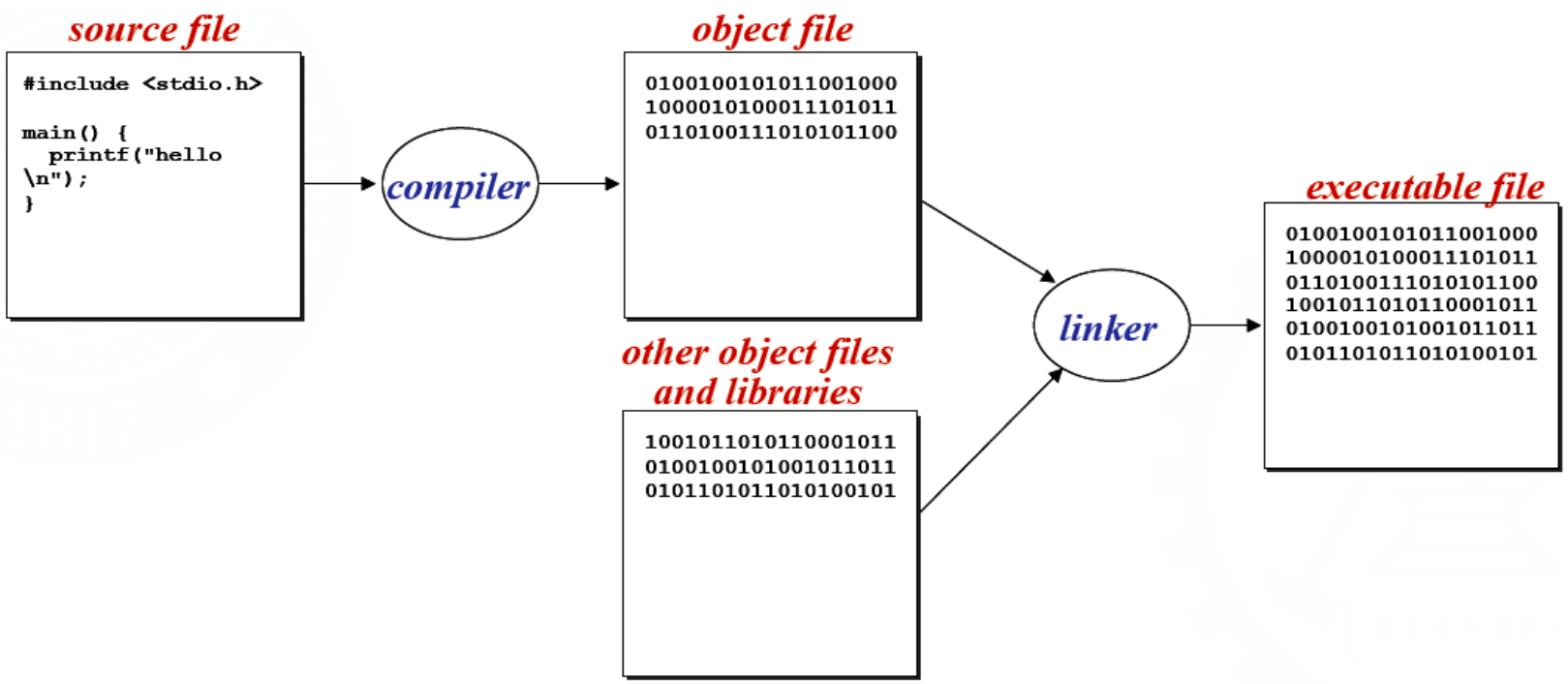
C/C++ History

- Developed in the 1970"s
  - Evolved from B language
  - Closely related to UNIX system
  - Inventor: Dennis Ritchie
- Designed for systems programming
  - Operating systems (such as Unix, Linux, …)
  - Utility programs, compilers
- Designed for memory-efficient programming
  - Designed to work in one pass
  - Memory access via pointers
  - Bit accessible (necessary for hardware operation)
- C++ is an Object Oriented Extension of C

Why C/C++?

- One of the most widely accepted programming languages
  - The most commonly-used language for embedded systems
- A “must know” tool for all engineers
- Mid-level language with high-level features (support functions and modules) and low-level features (hardware access via pointers)
- Very little time and space overhead – very efficient
- Very well understood
  - Can be easily understood by C++ or Java programmers
- Good and well-proven compilers
- Experienced staff and rich learning resources are available
- Fairly portable: compilers exist for virtually every processor

General Process of Programming in C/C++



Hello, World!

```
/****************************
 * First C program, hello.c *
 ****************************/
#include <stdio.h>

int main()
{
  printf("Hello, world!\n");
  return 0;
}
```

Comments!!

Library inclusion

Header file, for sharable definitions

C program is a collection of functions (a sequence of function calls), `main()` is the starting point of the sequence.

I/O function, not defined in C

Return from function calls to host environment (normally operating systems)

More about Hello World

- What it has?
  - A line of comment
  - A line of preprocessor directive
  - A function definition: main
  - An I/O statement
  - A return statement
- What it does?
  - Ask the computer to say hello to the world
  - It seems not computing, does it?
  - It seems not interacting with hardware, does it?

More about main() Function

- The main function is the entry point for C programs
- Every C program must have one and only one main function
- The main function has many variations
- Example:
  - main() {…}
  - int main() {… return 0;}
  - int main(argc, argv) {… return 0;}
- The curly brackets {…} enclose the function body

More About Library Inclusion

- A library is a collection of tools written by other programmers that perform specific operations
- Usually, you can “include” the header file of a library in your program
- stdio is standard input/output library predefined in ANSI C
  - printf() is one of the functions defined in this library
  - There are many other functions in this library
  - There are many libraries predefined in C standard or by users
  - stdio.h is the header file of the library stdio
- To include standard libraries in ANSI C:
  - #include <LibraryName.h>
- To include a user defined library:
  - #include “LibraryName.h”

Important Concepts in Programing

- Variable
- Data Structures
- Class (Object Oriented Programming)
- Operators
- Program Flow Control Statements
- Functions
- Algorithm

Variables

- A variable must have a *type, a name and a value*
- Variable name convention:
  - The name must start with a letter or the underscore
  - All other characters must be letter, digits, or the underscore
  - No space or other special characters are permitted
  - The name must not be a keyword (page 39, ASC)
  - C is case sensitive.
- Variables must be declared before use
  - inside function
  - outside function
  - in definition of function parameters

Variables

- Inside function – local variables
  - must be declared before used
  - declared at very beginning of a function (not true for C99)
  - variable local to a block {…}
  - local variables are located on stack or compiler created heap
- Example

```
void func1(void){
  int x;
  x = 10;
}

void func2(void){
  int x;
  x = -199;
}
```

different

Basic Data Type: String

- String is a null-terminated array of characters
  - null character: ‘\0’ has ASCII value of 0
  - a string of characters always has one more character –‘\0’
  - example: “hello” has 6 characters, is a string *constant*
- String initialization
  - char string_name[size] = “string”;
  - char str[9] = “I like C”; //‘\0’is automatically added
  - char str[9] = {‘I’,‘ ’,‘l’,‘i’,‘k’,‘e’,‘ ’,‘C’,‘\0’};
  - unsized string
  - char str[] = “I like C”;
  - /*array size is automatically calculated by compiler*/

Basic Data Type: Single-Dimension Arrays

- type array_name[size]
- Collection of multiple values of the same type
  - int test_score[10];
  - A collection of 10 integers
  - Array index: 0, 1, 2, … 9, higher index in higher memory address
  - Access individual element: test_score[2]
  - Cause memory to be allocated when array size is specified
  - Array is put in a contiguous block of memory
  - total bytes = sizeof(type of array) ×sizeof(array)
- Two cases when size of array need not be specified
  - size is implied by list of initial values
  - memory needs not to be allocated yet

Basic Data Type: Pointers

- Most powerful and dangerous feature of C
  - provide means to change function parameters –pass by reference
  - improve efficiency
  - support data structure
- Pointer is a variable that holds a memory address
  - type *name; defines pointer to object of certain type
  - Type could be any legal type
  - Type tells compiler what's being pointed to, but the actual type of the object could be something else
  - name = &var_1;generates an address of variable var_1
  - var_2 = *name; copy content of memory location pointed by name to var_2

Combined Data: Struct and Class

- C++ supports a product type: the struct
- This struct contains a multiple data types such as triangle's edge lengths:

```
struct Triangle {
        double a,b,c; //edge lengths
};
```

- When adding functions to data structures, we obtain classes.

Combined Data: Struct and Class

struct

- Heterogeneous aggregate data type
- **C style**
- **Contains only data**
- **Undefined by default**
- **All data is accessible**

class

- Heterogeneous aggregate data type
- **C++ style**
- **Contains data and functions**
- **Constructors can be used to initialize**
- **Control of data access**

Class

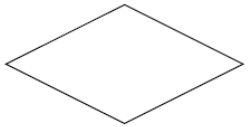
- A very powerful tool in C++ for complex tasks
- Widely employed in object oriented programming technique
- Provides great flexibility for abstraction and information protection
- Define a set of variables and functions as modular components
- Allow inheritance (from multiple parents) to reuse code
- Create object to instantiate a class entity
- Define virtual function to create template
- Include public, private and protected member status management
- Utilize overload and override of function for different behaviors
- Create constructor and destructor to initialize and clean up

Operators

precedence	operators	associativity
highest	`() [] -> .`	left to right
	`!~ ++ -- - (type cast) * & sizeof`	right to left
	`* / %`	left to right
	`+ -`	left to right
	`<< >>`	left to right
	`< <= > >=`	left to right
	`== !=`	left to right
	`&`	left to right
	`^`	left to right
	`|`	left to right
	`&&`	left to right
	`||`	left to right
	`?:`	right to left
	`= += ~= *= /= %= &= ^= |= <<= >>=`	right to left
	`,`	left to right

Program Flow Control

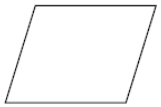
- Determines how the code executes to perform tasks
- Create flowchart before implementation
- Write Pseudo-code before coding
- Flow chart symbols:



Decision box

Program flow arrow

Action Box



Input/Output box

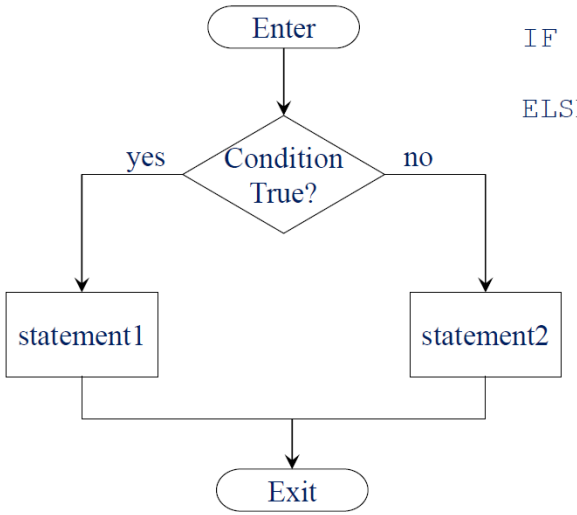
Off-page connector

Program terminator

Pseudo Code

- Besides flowchart, program solution can also be described using pseudo code
- Pseudo code is “sort of” a computer language, it mimics the syntax of Pascal or C, while using natural language in describing actions
- It facilitates the transition from a sketch of a solution to a programming language
- Independent of programming languages

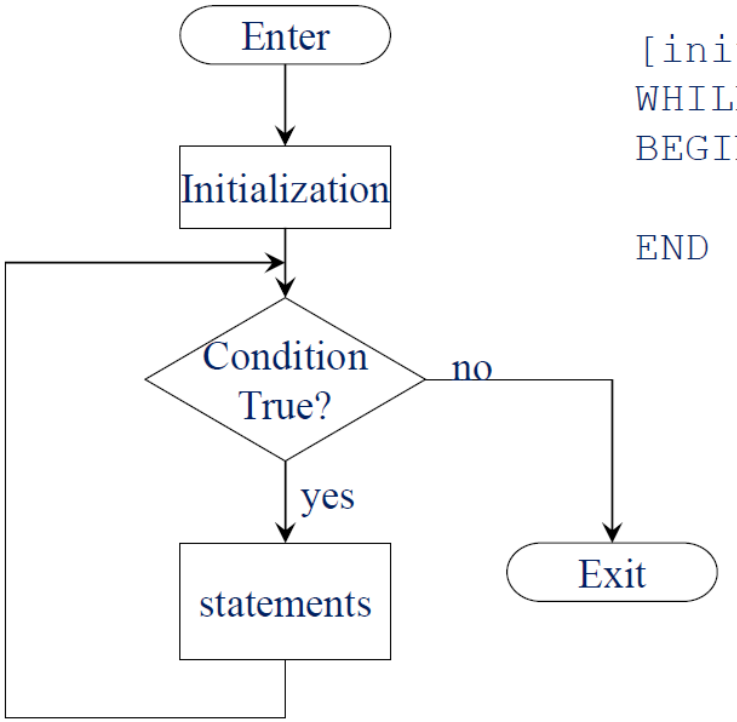
If… Else… Statement




```
IF [Condition]
      THEN [Statement 1]
ELSE
      [Statement 2]
```

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
int magic; /* magic number */
int guess; /* user's guess */
magic = rand(); /* generate the magic number */
printf("Guess the magic number: ");
scanf("%d", &guess);
if(guess == magic) {
printf("** Right ** ");
printf("%d is the magic number", magic);
}
else if(guess > magic)
printf("Wrong, too high");
else printf("Wrong, too low");
return 0;}
```

For Loop and While Loop




```
[initialization statements]
WHILE [condition] DO
BEGIN
      [statements]
END
```

```c
#include <stdio.h>
void main(void) {
    int x, neg_x;
    for(x=1; x <= 100; x++) {
        neg_x = -x;
        printf("%d\n", neg_x);
    }
}
```

```c
#include <stdio.h>
void main(void) {
    int x, neg_x;
    x=1;
    while(x <= 100) {
        neg_x = -x;
        printf("%d\n", neg_x);
        x++
    }
}
```

Functions

- A C function can be considered a black box
- Black box is usually a complicated electronic device that functions and is packaged as a unit and whose internal mechanism is usually hidden from or mysterious to the user.
- Broadly, black box is anything that has mysterious or unknown internal functions or mechanisms.
- Black box is an abstraction that communicates with its environment via well-defined inputs and outputs.

Why Functions?

- Function reduces the conceptual complexity of programs
  - Large programming problem divided into manageable pieces
- Function makes smaller functional unit reusable
- Function improves readability of program
- Function hides complicated implementation from applications
- Acceptable downsides
  - Too many function calls take extra time and slow down the program execution
  - Function calls take extra memory space to store information

Function Definition

- Functions must be declared before used
- Function declaration is called prototype of a function
- Once a function is declared as prototype, it can be called as many times as needed in following statements without actual definition
- Function must be defined (implemented) eventually In the same file
- In a different file of a project
  - In a library, e.g.#include <stdio.h>
  - printf(), scanf(), …
  - #include <math.h>
  - sqrt, sin, cos, …
- Use function by calling function name and provide input parameters

```
return_type function_name(parameter list)
{
  body of function
}
```

Algorithm

- Algorithm is an organized means to construct one solution of a problem, structured as a well-defined set of steps that can be carried out by a mechanism such as a computer
- Algorithm must be:
  - Clear, unambiguous, and effective with its steps being executable
  - Finite, in the sense that it terminates after a bounded number of steps
- Algorithm is the abstract strategy of solving a problem
- Function is the concrete realization of an algorithm in the context of programming language
- Developing algorithm before writing function
  - Might not seem to be necessary for simply problems
  - As problem becomes more complicated, algorithm development is a more organized manner of synthesizing thoughts and strategies

Arduino Code Structure

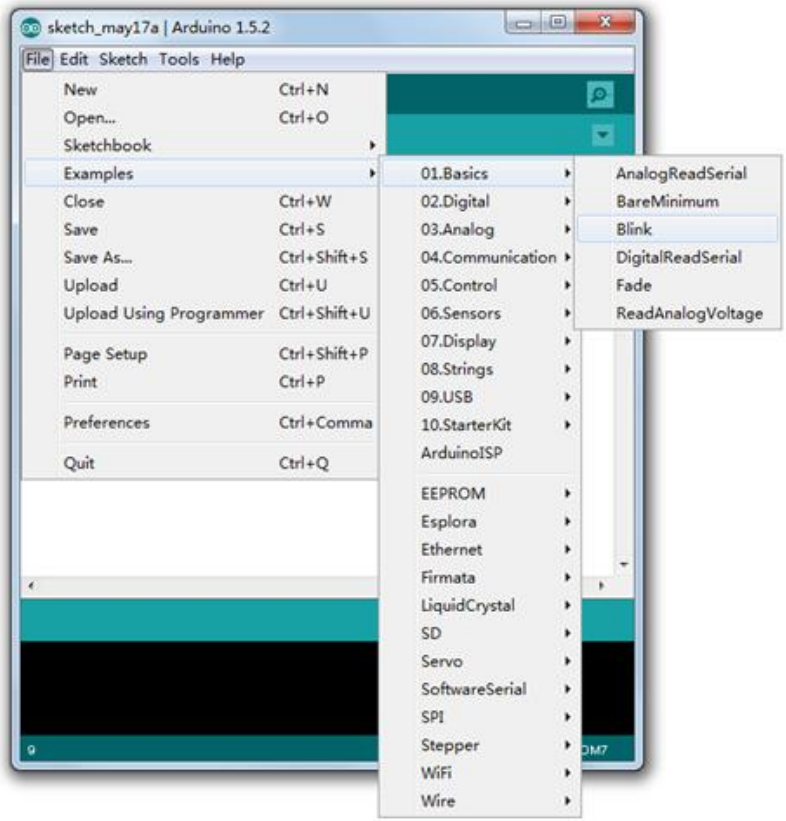
- Key Programming Language: C/C++
- Arduino Specific Constant Definition
  - Define Digital Pin Level: HIGH(1) | LOW(0)
  - Define direction of Digital Pins: INPUT | OUTPUT
- Code General Structure
  - void setup() and void loop() must exist for all Arduino program
  - void setup(): Initializing variables and modes of Pins
  - void loop(): Run program inside this function continuously

Arduino Basic IO Functions

- Function for Digital I/O
  - void pinMode(pin, mode): For defining the Pin.
    pin: 0~13; mode: INPUT | OUTPUT
  - void digitalWrite(pin, value): For output electric potential value.
    pin: 0~13; value: HIGH | LOW
  - int digitalRead(pin): For input electric potential value.
    pin: 0~13; value: HIGH | LOW
- Function for Analog I/O
  - int analogRead(pin): For reading analog signal. pin: A0~A5;
  - void analogWrite(pin, value) ~ PWM
- Function(Time function):
  - void delay(ms): For keeping running the program; Time Unit: ms.
  - void delayMicroseconds(us): For same purpose; Time Unit: us.

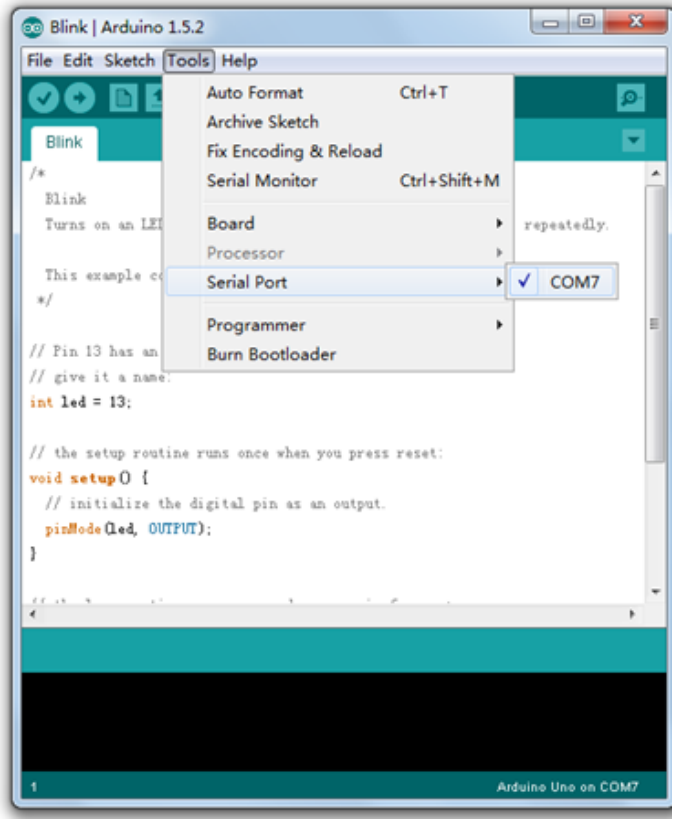
Arduino Quick Launch

- Open Arduino IDE
- Select File
- Go to Basic and Blink
- Click to Open



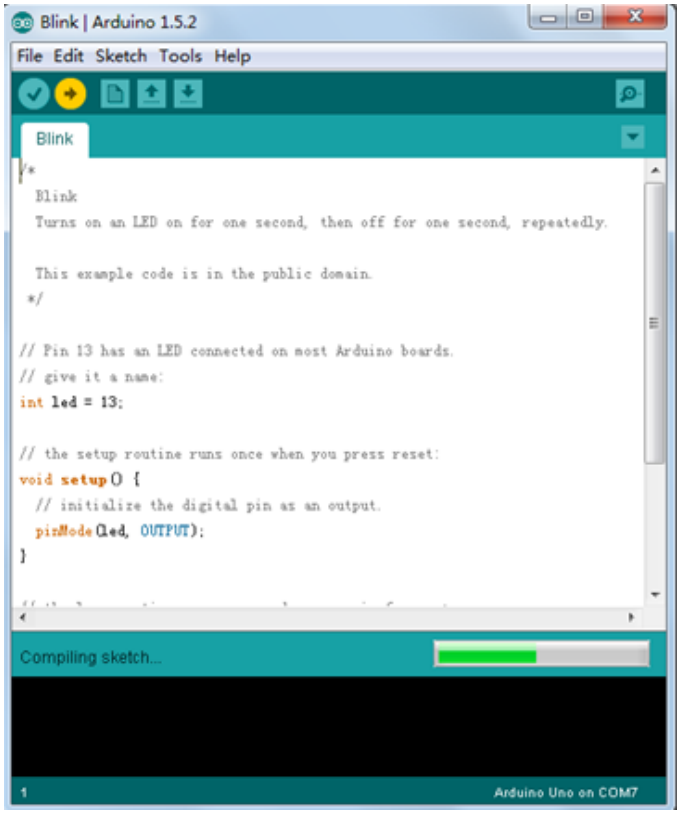
Arduino Quick Launch

- Plug in Arduino Uno
- Check for Available Serial Port
- Select the Port for Arduino



Arduino Quick Launch

- Click Tick to Compile Code
- Click Arrow to Upload Code
- View Info Area for Progress



Arduino Quick Launch

- Check “Done Uploading” for Status
- Watch the LED on Arduino Blink

Arduino Quick Launch

- Check “Done Uploading” for Status
- Watch the LED on Arduino Blink

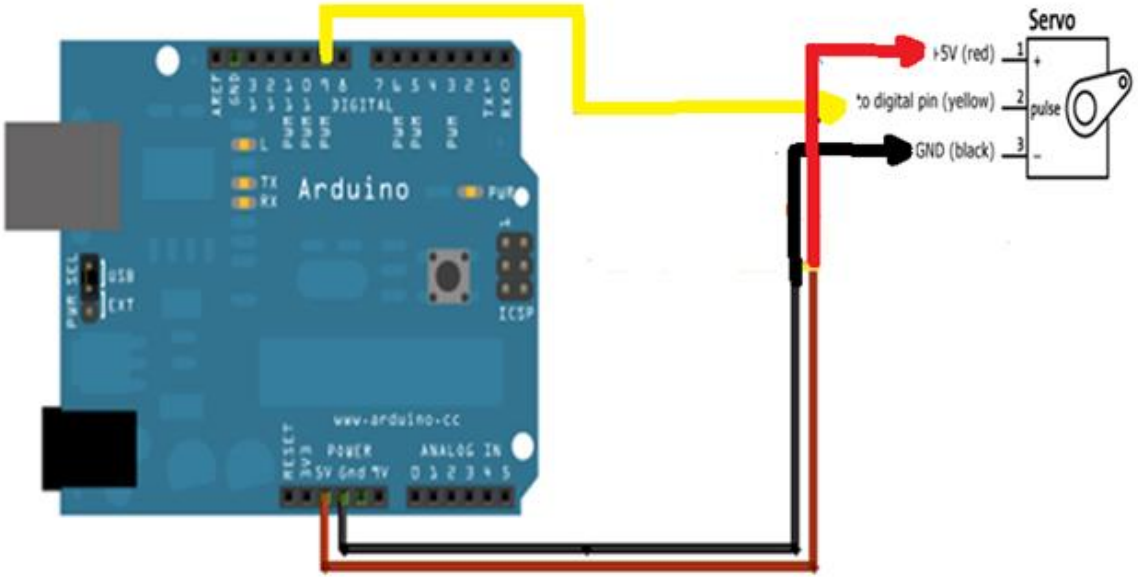
Serial Communication

- **Serial Communication Through Pin RX & TX:**

```
void setup(){
        Serial.begin(9600);
}

void loop(){
        Serial.println("Hello World！ ");
        delay(1000);
}
```

Control the Servo

- Connect the servo as shown
- Utilize example code “Servo Sweep” or the code on next page
- Try modify the code for controlling the gripper



Servo Code

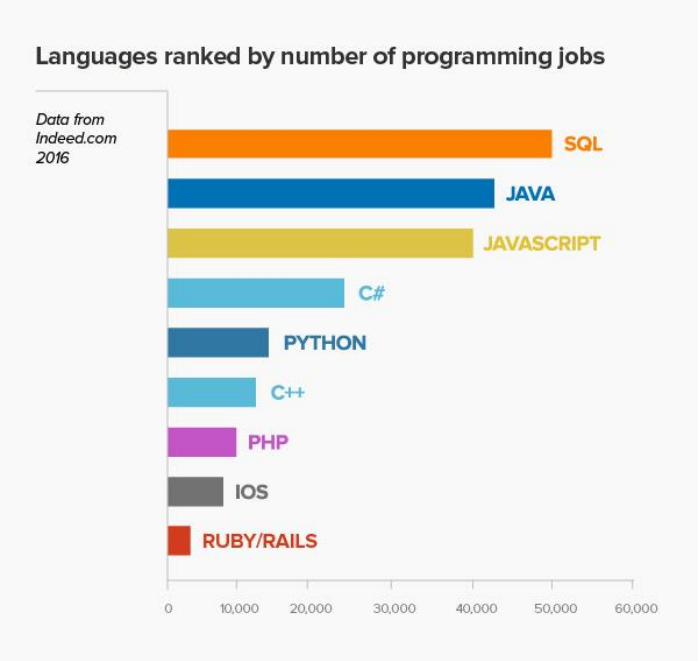
```
//this code will make it so when you upload it the servo will
turn to the //right then to the left

#include <Servo.h>
Servo servoMain;
void setup(){
        servoMain.attach(10);
}
void loop() {
        servoMain.write(180);
        delay(1000);
        servoMain.write(0);
        delay(1000);
}
```

Introduction to Python

- Python was started in late 1989 by Guido van Rossum
- It is named after The Monty Python
- He wanted a user-friendly coding language that could be ported to many different applications
- In February 1991 Van Rossum's code was posted to USENET where it became an open-source language that is consistently updated

Coding Languages

- There are hundreds of programming languages
- Some languages are application specific, but relatively simple:
  - R for Statistics
  - SQL for database building
  - VBA for in Office use
- Others can do many applications but are complicated:
  - C,C#,C++
- Some are a great mix:
  - Python!



What is Python good for

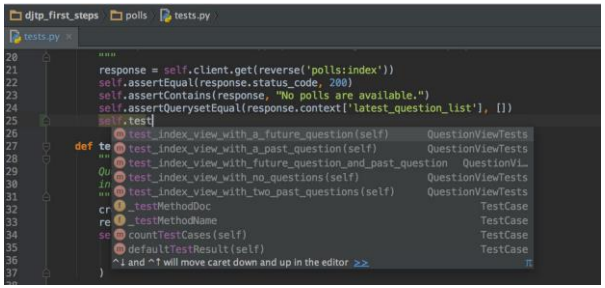
- Python is a high-level general-purpose programming language that can be applied to many different classes of problems
  - String processing (regular expressions, Unicode, calculating differences between files)
  - Internet protocols (HTTP, FTP, SMTP, XML-RPC, POP, IMAP, CGI programming)
  - software engineering (unit testing, logging, profiling, parsing Python code)
  - operating system interfaces (system calls, filesystems, TCP/IP sockets)
- All above from python.org directly

Python 2.7 vs. 3

- We are using Python 3.5. Many users still code in Python 2.7 even though Python 3 was released in 2008
- This is largely due to the large number of Python 2.7 libraries that were not converted over to Python 3
- Syntax differences (won’t discuss them here)

Python IDEs

- Integrated Development Environments (IDEs) are the tools used to write code
- They provide visuals and tools to organize and debug code.
- Python has several IDEs:
  - IDLE (named after Eric Idle from The Monty Python)
  - Pycharms
  - PyDev

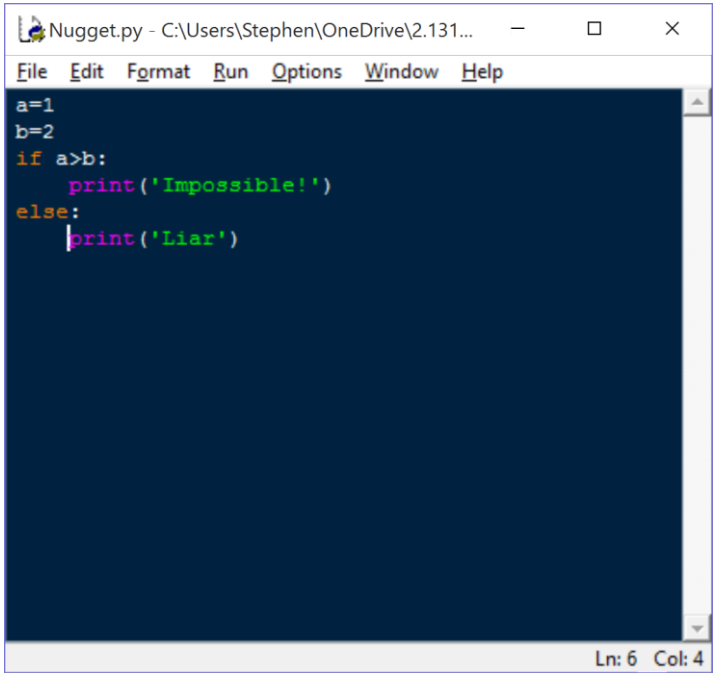


Advantages of Python

- You can write programs much faster in Python
  - “3-5x faster than Java”
  - “5-10x faster than C++”
- Easier to learn
  - Used at companies like Google and Facebook
- Large selection of libraries for varying applications

Let get started!

- Open Python IDLE 3.5 (You should have this downloaded already)
- Python Shell
  - For quick tests, does not save code
  - a=5 , b=4 , a+b=?
- Python Editor
  - For code that you want to repeat



Type of Variables

- Integers – 10
- Floats – 10.0
- Strings – ‘10’
- Boolean - TRUE
- Lists – [10.0,11.0,12.0]
- Dictionary – [‘ten’:’10.0’,]

- Neat thing in Python: you don’t need to declare variables

Integers/Floats

- Integers
  - Whole Numbers
  - Int() – translate variable into an integer

```
>>> a = 5
>>> type(a)
<class 'int'>
>>> a
5
```

- Floats
  - Has a decimal
  - Float() – translate variable into a float

```
>>> a = 5.0
>>> type(a)
<class 'float'>
>>> a
5.0
```

Math Operations

- Addition: +
- Subtraction: -
- Multiplication: *
- Division: /
- Whole number Division: //
- Exponent: **
- Here is a good cheat sheet: http://learnpythonthehardway.org/book/ex37.html

Comparisons

- ==
  Equal to
- >, >=
  Greater than (or equal)
- < , <=
  Less than (or equal)
- !=
  Not equal
- And
- Or
- Not

```
>>> not 1==2
True
>>> 4 and 2 >3
False
>>> 4 and 5 >3
True
```

Boolean

- TRUE or FALSE
- Example: strings in string

```
>>> for each in a:
        each == 4


False
False
False
True
False
False
False
```

```
>>> a
[1, 2, 3, 4, 5, 6, 7]
>>> 3 in a
True
>>> 8 in a
False
```

Built-in Functions

- There are some functions ready to go!

- Type dir(__builtins__) - example
- List of functions is available
- Type help(max)

- Let’s try max and min

Functions

- If you need to do something more than once, write a function
- Must start with def
- End with a return, or executes a function that ends in a return
- Everything that happens *within* a function (except for the output) is forgotten once the function is completed

```
>>> def square(x):
        return x**2

>>> square(4)
16
```

Strings

- A sequence of characters
  - Characters must be surrounded by ‘ or “
  - Can be added to other strings
  - Can be multiplied by floats or integers
  - Str() – translate variable into a string
  - Len() – return length of string
  - Indexing
    - String[0] – return first character in String
    - String[0:3] – return first three characters in String
    - String[-2] – return the second to last character

Lists

- A = [1,2,3,4]
- Non-modifiers
  - List[0] – just like a string
  - Max(), min(), sum(), len()
  - List.count(value), list.index()
- Modifiers
  - list.append(*value*)
  - list.extend(list)
  - list.sort()
  - list.insert(index,value)
- Other
  - Range(int)

Inputs/Outputs

- Input(‘how old are you?’)

```
>>> name = input('What is your name?: ')
What is your name?: Stephen
>>> name
'Stephen '
```

- Print(variable)

```
>>> print('this is how you print')
this is how you print
```

If Statements

If argument:
    action
elif argument:
    action
else:
    other action

```
a=1
b=2
if a>b:
    print('Impossible!')
else:
    print('Liar')
```

- Unlike languages like C, Python uses indents to signify hierarchy

For Loops

- for element in array:
  action
- Range() is often used

- Indents are again important

```
>>> for each in 'Stephen':
        print('the letter is ', each)


the letter is  S
the letter is  t
the letter is  e
the letter is  p
```

Libraries

- The wheel has already been invented, you don't have to invent it
- Likewise, many functions in Python have already been coded, so you can just download them and use them right away
- Libraries include:
  - BeautifulSoup, Scrapy – Web Scraping
  - NumPy, SciPy – Advanced Math
  - Matplotlib – Numerical Plotting
  - Pillow – Image Processing

Additional Resources

- Look into additional examples in the development environment
- Check out the Arduino Forum: https://forum.arduino.cc/
- Check out projects on Instructable: http://www.instructables.com/
- Learn Python at: http://www.learnpython.org/

Homework

- Write 3 functions and execute them in the shell or run in a main function all together
  - The addition of two equally sized lists (provide error messages specifying which list is bigger if the user enters unequally sized lists) – **add_list(list1,list2)**
  - Counts the number of vowels in a string (upper and lower case). The number of vowels should be returned, and the vowels counted should be printed – **vowel_count(string)**
  - Pythagoreans Theorem using the *input()* function to enter the legs of the triangle – **pythag()**

MITMECHE

Thank You!