NYU K12 STEM Internet of Things 2019

WELCOME

NYU TANDON SCHOOL OF ENGINEERING

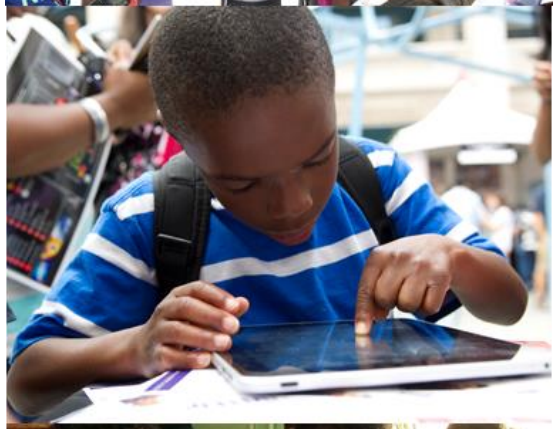
Center for K12 STEM Education

Center for K12 STEM Education

NYU TANDON SCHOOL OF ENGINEERING

K12 STEM Mission

- Access and opportunity
- Design and deliver high-quality, innovative teaching and learning programs for teachers and students
- Connect to faculty research
- Support and develop our students
- Create in-depth STEM education partnerships



Schedule

- Class hours: 9am to 4pm
  - ML is in Room RH 304
  - SPARC is in Room RH 317
  - IoT is in Room RH 707

- University Programs Orientation will be held **Sunday June 16, July 7, July 28 3:00pm. The 16th and the 7th will be in the MakerSpace. July 28th will be in the SPARC classroom**.

- International Students must check in with OGS at **5 MetroTech RM: 259 at 3pm on these dates**:
  - **Friday, June 21**
  - **Monday, July 8**
  - **Monday, July 29**

- **Admissions 101 will take place every second Monday June 24, July 15, August 5 at Wunsch Hall at 4:00pm**

TUTOR DETAILS

- **Contact Details:**

- Monish N. Kapadia
  mnk337@nyu.edu
- Milind Singh
  ms10997@nyu.edu
- Ishita Choudhary
  ivc211@nyu.edu
- Tara Nicole Umesh
  tnu201@nyu.edu

- **Program Coordinator:**
- Mathhew S Campisi
  msc398@nyu.edu

CLASS RULES

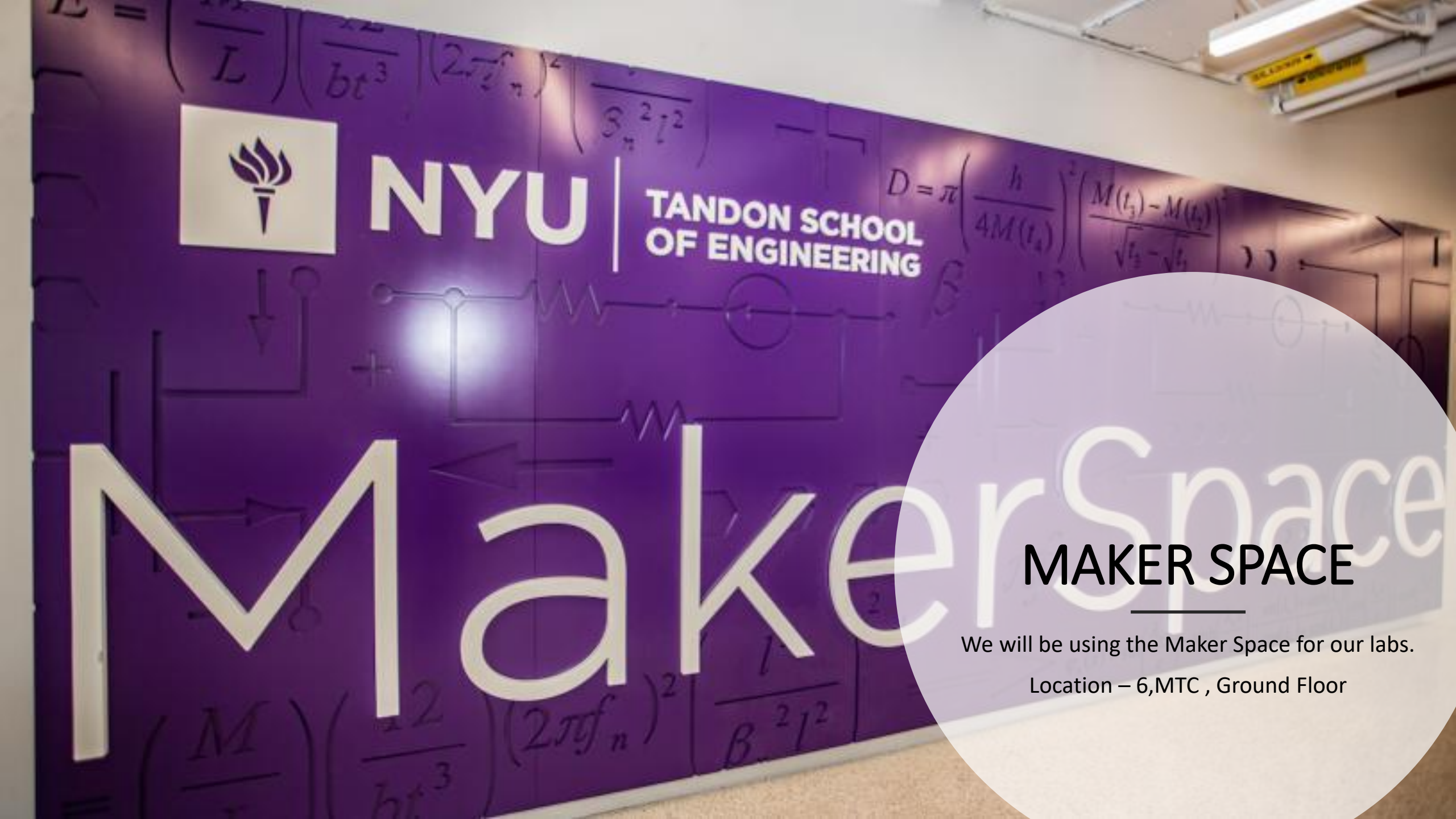
- ASK AWAY ANYTHING.
- Explore what interests you , ask for extra material.
- Instructors will be your best friends in getting through this course.
- Accept Challenges , Try , Fail , Don’t give up , TRY AGAIN.
- Maintain professional attitude during class , deviant behavior will be recorded and reported.
- Attendance will be recorded and reported.
- We will keep track of your progress through grades on your labs and you will be awarded a grade letter at the end of course.

CLASS STRUCTURE

- Class Hours : Discussions on Concepts , Demonstrations
- Lab Hours : Practice through in-class assignments.
- In-Case a project is not completed in class , it is to be submitted next day at the start of the class.
- Submission Instructions TBD.
- Two Arrangements Possible:
  - 9AM-12PM → Class Hours
  - 1PM-4PM → Lab Hours
  - Alternate hours between labs and classes.

GIT-HUB

- We have a GitHub set up for this program.
- The link is https://github.com/milli9d/NYUSTEMIoT2019
- You can use the Git to access all code discussed in class as well as class study materials.
- Submit you e-mail IDs before the end of class.



MAKER SPACE

We will be using the Maker Space for our labs.

Location – 6,MTC , Ground Floor

Internet Of Things

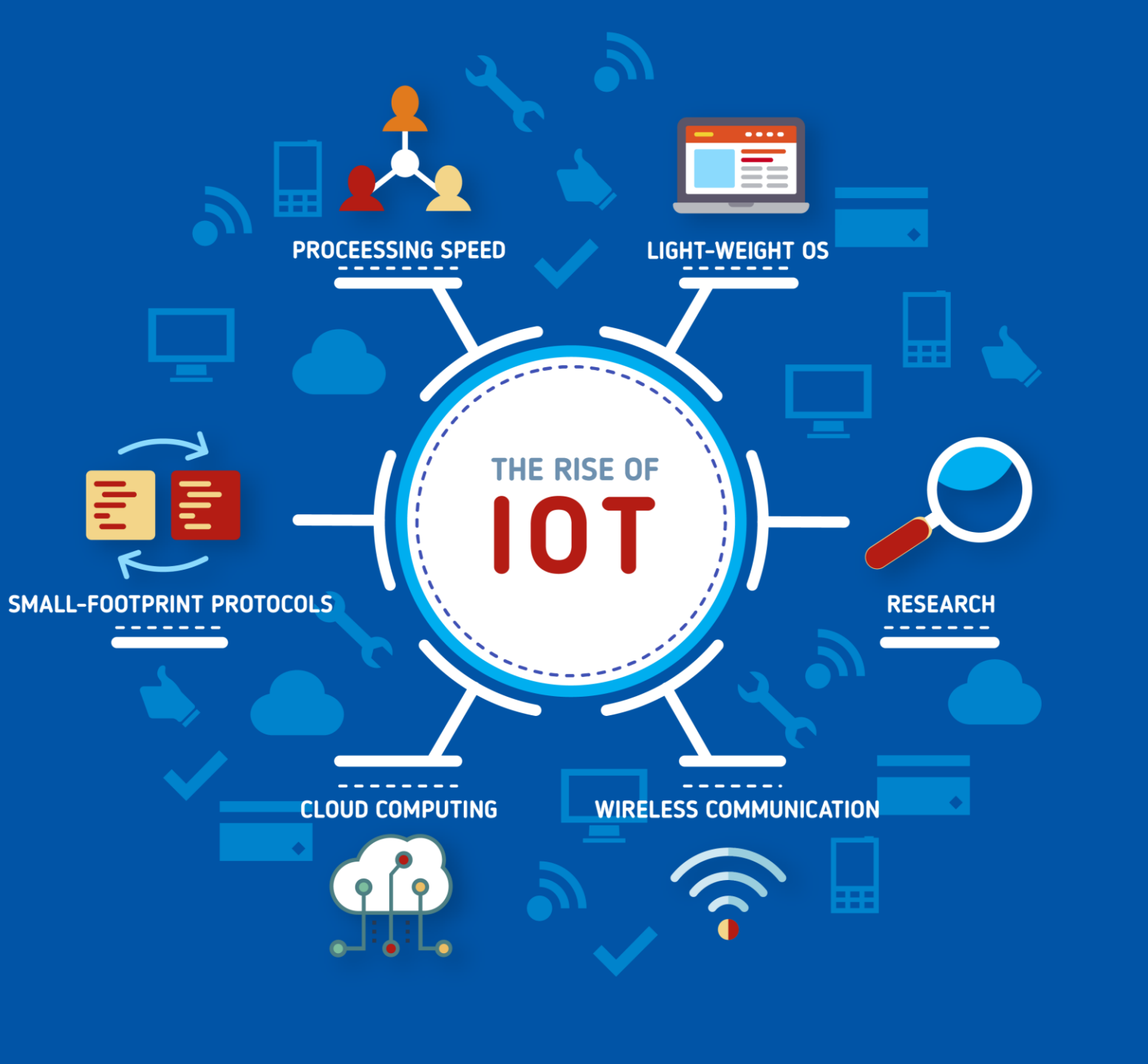
What are Embedded Systems?

- An embedded system is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints. It is **embedded** as part of a complete device often including hardware and mechanical parts. Embedded systems control many devices in common use today.
- Anything that uses a microprocessor but isn't a general-purpose computer

  Smartphones, Set-top boxes ,Televisions ,Video Games ,Refrigerators ,Cars ,Planes ,Elevators ,Remote Controls ,Alarm Systems
- The user "sees" a smart (*special-purpose*) system as opposed to the computer inside the systems
  - "how does it do that?"
  - "it has a computer inside of it"
  - "oh, BTW, it does not or cannot run Windows or MacOS"
- The end-user typically does not or cannot modify or upgrade the internals

Why Is IoT Significant?

- The **Internet of things** (**IoT**) is the extension of Internet connectivity into physical devices and everyday objects.
- Embedded Systems have evolved into a new discipline called IoT.
- IoT emphasizes on creating networks of large number of smaller embedded devices which can be remotely monitored and controlled.
- IoT is now also extending to Personal Mobile Devices , where researchers are using huge numbers of interconnected PMDs to extract parallel computing without the need of dedicated hardware.
- IoT is also crucial in big data applications ,such as city planning , architecture , Departmental Stores , etc.



Embedded Systems Scope

Automotive systems

- Perhaps designing and developing “drive-by-wire” systems
- Self-driving vehicles

Telecommunications

Medical Devices

Consumer electronics

- Cellular phones, MP3 devices, integrated cellular/tablet
- Set-top box and HDTV
- Home and Internet appliances/ IOT
- Your refrigerator will be on the internet more than you are!

Defense and weapons systems

Process control

- Gasoline processing, chemical refinement

Automated manufacturing

- Supervisory Control and Data Acquisition (SCADA)

Space communications

- Satellite communications

Why are Embedded Systems Different?

Four General Categories of Embedded Systems

1. General Computing
   - Applications similar to desktop computing, but in an embeddedpackage
   - Video games, set-top boxes, wearable computers, automatic tellers
   - Tablets, Phablets
2. Control Systems
   - Closed loop feedback control of real-time system
   - Vehicle engines, chemical processes, nuclear power, flight control
3. Signal Processing
   - Computations involving large data streams
   - Radar, Sonar, Video compression
4. Communication & Networking

   Switching and information transmission

   Telephone system, Internet

   Wireless everything- IoT

Real-Time Embedded Systems

- These systems are designed to react to real-time stimulus , either external or internal.
- Precise timing and accuracy are required.

Typical Embedded Systems Constraints

- Small Size, Low Weight
  - Handheld electronics
  - Transportation applications weight costs money
- Low Power
  - Battery power for 8+ hours (laptops often last only 2 hours)
  - Limited cooling may limit power even if AC power available
- Harsh environment
  - Heat, vibration, shock
  - Power fluctuations, RF interference, lightning
  - Water, corrosion, physical abuse
- Safety critical operation
  - Must function correctly
  - Must not function incorrectly
- Extreme cost sensitivity
  - $0.05 adds up over 1,000,000 units

Embedded Systems Designer Skill Set

- Appreciation for multidisciplinary nature of design
  - Both hardware & software skills
  - Understanding of engineering beyond digital logic
  - Ability to take project from specification through production
- Communication and Teamwork skills
  - Work with other disciplines, manufacturing, marketing
  - Work with customers to understand the real problem being solved
  - Make a good presentation; even better – write “trade rag” articles
- And, by the way, technical skills too…..
  - Low-level: Microcontrollers, FPGA/ASIC, assembly language, A/D, D/A
  - High-Level: Object oriented design, C/C++, Real Time Operating Systems
  - Meta-level: Creative solutions to highly constrained problems
  - Likely in the future: Unified Modeling Language, embedded networks

Intro To Programming

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    printf("!!!Hello World!!!"); /* prints !!!Hello World!!! */
    return 0;
}

```

Hello World

Data types

- Used by compiler to determine what kind of data is being stored.
- Data can be a number, word, character, etc.

Data types
Numbers

- Storing integers
  - Syntax: *int variableName*
  - Eg. *int number = 10;*
  - *Max Value: 2,147,483,647*
- *Storing a large number*
  - Syntax: *long variableName*
  - *long bigNumber = 9999999999999;*
- *Storing decimals*
  - Syntax: *float variableName*
  - Eg. *float marks = 89.5;*
  - *Precision: 6 to 9 digits*
- *Storing higher precision number*
  - Syntax: *double variableName*
  - Eg. *double pi = 3.1415926535;*

Data types

Words/Characters

- Storing characters

  Syntax: *char variableName*

  Eg.char alphabet = “A”;

- *Storing string*

  Syntax: *char variableName[]*

  Eg. char name[50] = “IoT Camp”;

Data types

Boolean

- Boolean data types are used to represent the truth values of a logic.
- It can have 2 values, either *true* or *false.*
- To include this data type, a library has to be added.

  #include <stdbool.h>
- Storing a Boolean

  Syntax: *bool variableName*

  Eg. bool isPlaying = true;

Printing on console

- Syntax: printf("*statement*",identifierValue);
- Different identifiers:
  - d – integers
  - x – hexadecimal
  - s – string
  - c - character
  - f - floating numbers
- Eg: char name[50] = "IoT";

  int year = 2019;

  printf("Welcome to Summer %s %d camp.", name, year);

  Output: `Welcome to Summer IoT 2019 camp.`
- Formatting the output:
  - \n – new line
  - \t – four horizontal spaces

- Eg: printf("Sr.no. \t Name \n 1. \t Jack");

  Output:

```
Sr.no.    Name
 1.       Jack
```

Taking input from user

- Syntax: scanf("*identifier*", &variableName);
- Eg: int number;

  scanf("%d", &number);
- Important point:

  While using a string to take an input DON'T use "&" i.e.

  char name[50];

  scanf("%s", name); // No & before name

```c

#include <stdio.h>
#include <stdlib.h>

int main(void) {
    setvbuf(stdout, NULL, _IONBF, 0);
    char name[50];
    printf("Enter your name: ");
    scanf("%s",name);
    printf("Hello %s", name);
    return 0;
}

```

Problems Tasks Console Properties

<terminated> (exit value: 0) helloWorld.exe [C/C++ Application

```
Enter your name: Monish
Hello Monish
```

Hello you

Comments

- Commenting a code is always a good practice.
- It'll help you as well as others to understand the code.
- Single line comment: `//Single line comment`
- Multiline comment:

```
/*
 * Multi line comment
 * for something long
 *
 */
```

Mathematical Operations

Operations	Syntax
Addition	number1 + number2
Subtraction	number1 - number2
Multiplication	number1 * number2
Division	number1 / number2
Remainder	number1 % number2

Important point:
In division, the result of the operation will be the quotient only i.e. it will not give the decimal value.
Eg. int number1 = 15;
int number2 = 10;
printf(“%f”,(number1 / number2));
Output: 1.000000 instead of 1.5
To obtain decimal value one of the variable has to be a float.
Eg. float number1 = 15;
int number2 = 10;
printf(“%f”,(number1 / number2));
Output: 1.50000

Switch case

- Syntax: switch(variable)

```
{
    case value:
        // do something
        break;
    case value2:
        // do something else
        break;
    default:
        // if none of the above case satisfy then do something else
        break;
}
```

Switch case

- Syntax: switch(variable)

```
{
    case value:
        // do something
        break;
    case value2:
        // do something else
        break;
    default:
        // if none of the above case satisfy then do something else
        break;
}
```

- Eg. int number = 1;

```
switch(number)
{
    case 0:
    printf("Case 0 selected");
    break;
    case 1:
    printf("Case 1 selected");
    break;
    default:
    printf("Wrong selection");
    break;
}
```

```c
#include <stdio.h>
#include <stdlib.h>


int main()
{
  char alphabet;
  printf("Select a character: ");
  scanf("%c", &alphabet);
  switch(alphabet)
  {
    case 'A':
    printf("You selected apples\n");
    break;
    case 'B':
    printf("You selected bananas\n");
    break;
    case 'G':
    printf("You selected grapes\n");
    break;
    case 'O':
    printf("You selected oranges\n");
    break;
    default:
    printf("No fruit selected\n");
    break;
  }
  return 0;
}
```

Switch case

Calculator

Design a calculator that performs the following operations:

```
Enter first number: 11
Enter second number: 25
 ------------------------------
|       1.| Addition           |
|       2.| Subtraction        |
|       3.| Multiplication     |
|       4.| Division           |
 ------------------------------
Enter operation number: 1
11 + 25 = 36
```

```c
int main(void) {
    setvbuf(stdout, NULL, _IONBF, 0);
    int firstNumber, secondNumber, operation;
    printf("Enter first number: ");
    scanf("%d", &firstNumber);
    printf("Enter second number: ");
    scanf("%d", &secondNumber);
    printf(" ------------------------------\n");
    printf("|   1.| Addition         |\n");
    printf("|   2.| Subtraction      |\n");
    printf("|   3.| Multiplication   |\n");
    printf("|   4.| Division         |\n");
    printf(" ------------------------------\n");
    printf("Enter operation number: ");
    scanf("%d", &operation);
    // printf("\n");
    switch(operation)
    {
        case 1:
            printf("%d + %d = %d",firstNumber,secondNumber,(firstNumber + secondNumber));
            break;
        case 2:
            printf("%d - %d = %d",firstNumber,secondNumber,(firstNumber - secondNumber));
            break;
        case 3:
            printf("%d * %d = %d",firstNumber,secondNumber,(firstNumber * secondNumber));
            break;
        case 4:
            printf("%d / %d = %d",firstNumber,secondNumber,(firstNumber / secondNumber));
            break;
        default:
            printf("Invalid operation number");
            break;
    }
    printf("\n");
    return 0;
}
```

Calculator