IEEE UCF

IEEE Project Competition

Week 2: Microcontrollers

UCF | College of Engineering and Computer Science

Reminders

- Please be an registered IEEE members by week 3! We will be check for it, if your not sure ask us any questions you have and we will help you.
- Groups aren’t 100% final today but I will be putting everyone in groups and if someone wants to join later on they must be a IEEE member.
- The MSP430’s you can’t keep (sorry!).
- Be careful with the MSP430’s and all electronics in general.

Meet the Team

IEEE UCF

Matias Guillen

Project Chair
Sophomore - Electrical Engineering
Internal Project Competition - Lead
Contact: vector8925

Erik Siegman

Project Committee
Sophomore - Computer Engineering
Internal Project Competition - Sub-Lead
Contact: eriks0527

Yousef Awad

Project Committee
Sophomore - Computer Engineering
Internal Project Competition - Sub-Lead
Contact: quil180

Meet the Team

Tino Hernandez

Project Committee
Sophomore - EE/CS/CPE/Delusion
Internal Project Competition - Sub-Lead
Contact: vjmuzik

This Meeting's Agenda

Overview of MSP430 microcontrollers

Setting up the Arduino IDE

Basic C++ programming concepts

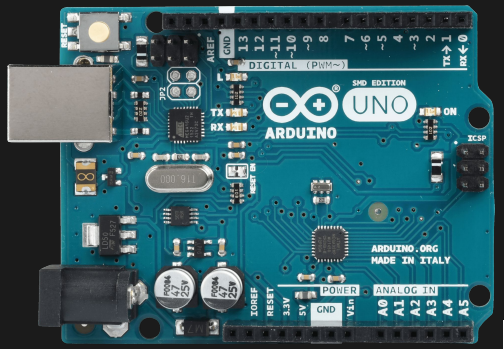
Interfacing with microcontroller pins

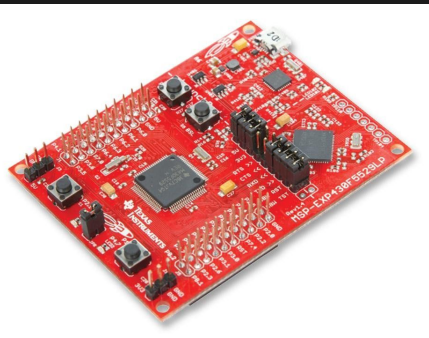
Demonstration of servo/motor control

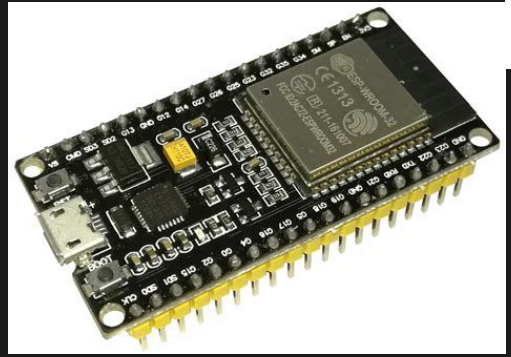
Finalize Teams

What is a Microcontroller

- Definition:
  - A microcontroller is a small, low-cost computer on a single integrated circuit (IC) that includes a processor, memory, and input/output (I/O) peripherals.
- Why Use Microcontrollers?
  - Control Systems: Commonly used in embedded systems for controlling specific functions.
  - Cost-Effective: Inexpensive solution for automation and small-scale computing.
  - Versatility: Widely used in consumer electronics, automotive, industrial control, robotics, and more.





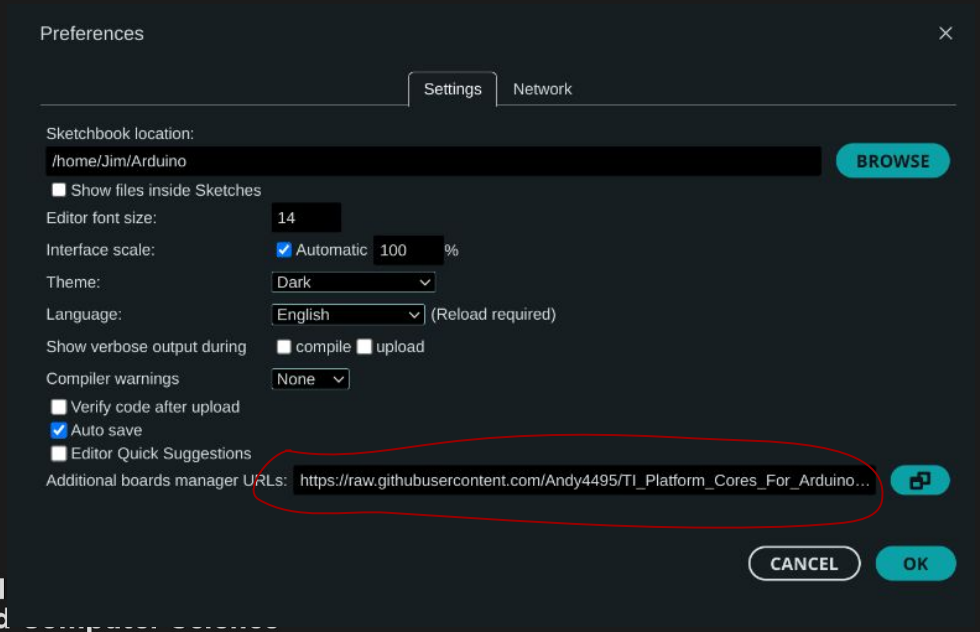


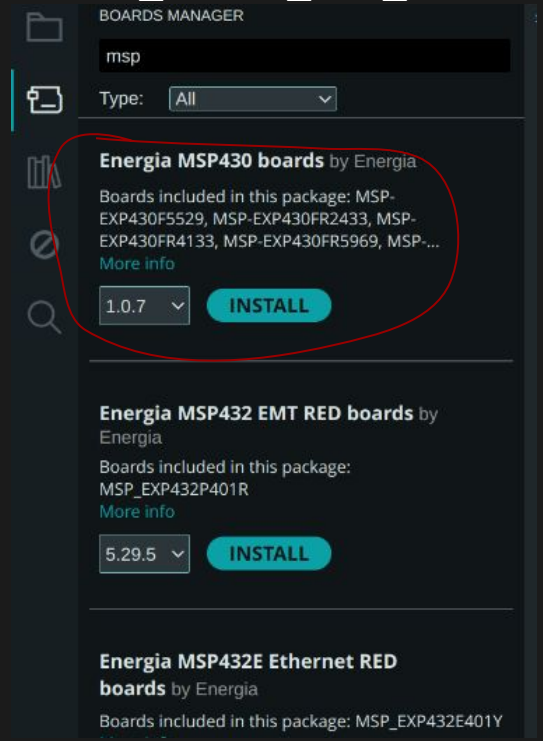
Arduino IDE

- Why Use Arduino IDE for MSP430?
  - User-Friendly Interface: Simple, intuitive environment for beginners and experienced users.
  - Cross-Platform Compatibility: Works on Windows, macOS, and Linux.
  - Community Support: Large user base with extensive tutorials and forums.
  - Integrated Tools: Built-in code editor, compiler, and serial monitor.
- Go to https://www.arduino.cc/en/software and download the latest version (2.3.2)

Arduino IDE

https://raw.githubusercontent.com/Andy4495/TI_Platform_Cores_For_Arduino/main/json/package_energia_optimized_index.json





Install Drivers

https://www.ti.com/tool/download/MSPDS-USB-DRIVERS/1.0.1.2

Make an account and download the proper drivers, or download the file in Discord

Open the file and follow the prompt to install

C++

- What is C++?
  - A powerful, general-purpose programming language.
  - Widely used for system programming, game development, embedded systems, and applications requiring high performance.
- C++ Basics
  - Variables: Named storage for data that your program can manipulate.
    - int - Integer numbers (e.g., int age = 21;)
    - float - Floating-point numbers (e.g., float temperature = 98.6;)
    - char - Single characters (e.g., char grade = 'A';)
    - bool - Boolean values (true or false) (e.g., bool isOpen = true;)
    - string - Sequence of characters (requires #include <string>) (e.g., string name = "Alice";)
  - Operators
    - Arithmetic Operators: +, -, *, /, % (modulus)
    - Comparison Operators: ==, !=, <, >, <=, >=
    - Logical Operators: && (and), || (or), ! (not)

C++

- Conditionals
  - ```
    if (condition) {
      // Code to execute if condition is true
    } else {
      // Code to execute if condition is false
    }
    ```

- Loops
  - For loop: repeats a specified number of times

    ```
    for (int i = 0; i < 5; i++) {
      // Code to execute
    }
    ```

  - While loop: repeats while a condition is true

    ```
    while (condition) {
      // Code to execute
    }
    ```
    ```

C++

- Functions: Reusable blocks of code designed to perform specific tasks.
  - Syntax:

```
returnType functionName(parameters) {
  // Code to execute
  return value; // Optional, if returnType is not void
}
```

  - Example:

```
int add(int a, int b) {
  return a + b;
}
```

C++ for Microcontrollers using Arduino

- How Arduino Programming Differs from Standard C++:
  - Simplified Setup: No need for a main() function; instead, use setup() and loop():
    - setup(): Runs once at the start to initialize settings (e.g., pin modes, serial communication).
    - loop(): Repeatedly executes as long as the board is powered.
  - Built-in Functions and Libraries:
    - Arduino provides built-in functions for hardware interaction (e.g., digitalRead(), digitalWrite(), analogRead(), delay()).
    - Many pre-written libraries make complex tasks simpler (e.g., controlling motors, handling communication protocols).

C++ for Microcontrollers using Arduino

- Key Functions:
  - setup() Function:
    - Purpose: Runs once when the microcontroller is powered on or reset.
    - Use: Initialize settings (e.g., setting pin modes, starting serial communication).
  - loop() Function:
    - Purpose: Runs continuously after setup() is complete.
    - Use: Main body of the program; repeatedly executes the code inside, allowing the microcontroller to perform tasks over and over.
  - pinMode(pin, mode) Function:
    - Purpose: Configures a specific pin to behave as either an input or an output.
    - Parameters:
      - pin: The number of the pin to set (e.g., 13).
      - mode: INPUT, OUTPUT, or INPUT_PULLUP.

C++ for Microcontrollers using Arduino

- Key Functions:
  - digitalWrite(pin, value) Function:
    - Purpose: Sends a HIGH or LOW signal to a digital pin.
    - Parameters:
      - pin: The number of the pin to write to (e.g., 13).
      - value: HIGH (turns the pin on) or LOW (turns the pin off).
  - digitalRead(Pin) Function:
    - Purpose: Reads the state of a specified digital input pin.
    - Parameters:
      - pin: The number of the digital pin you want to read (e.g., 2).
    - Returns:
      - HIGH (1) if the pin is receiving a voltage signal.
      - LOW (0) if the pin is not receiving a voltage signal.
  - delay(milliseconds) Function:
    - Purpose: Pauses the program for a specified amount of time.
    - Parameters:
      - Milliseconds: Time to wait in milliseconds (1000 milliseconds = 1 second).

Basic Arduino Program

- Key Points:
  - pinMode(ledPin, OUTPUT): Configures the LED pin as an output so it can send a signal.
  - digitalWrite(ledPin, HIGH): Sends a HIGH signal (5V or 3.3V depending on your board) to turn the LED on.
  - digitalWrite(ledPin, LOW): Sends a LOW signal (0V) to turn the LED off.
  - delay(1000): Pauses the program for 1000 milliseconds (1 second) between turning the LED on and off.
- What You'll See:
  - The LED connected to pin 13 will blink on and off, with a 1-second interval between each state change.

```
// Define the pin where the LED is connected
const int ledPin = 13;  // LED connected to digital pin 13

// The setup function runs once when you press reset or power the board
void setup() {
  pinMode(ledPin, OUTPUT);  // Set the LED pin as an output
}

// The loop function runs over and over again forever
void loop() {
  digitalWrite(ledPin, HIGH);  // Turn the LED on (HIGH voltage level)
  delay(1000);                 // Wait for 1 second (1000 milliseconds)
  digitalWrite(ledPin, LOW);   // Turn the LED off (LOW voltage level)
  delay(1000);                 // Wait for 1 second
}
```

Running an Arduino Program

- Connect your MSP430 to the computer

1) Select the board from the dropdown
2) Use to check if code will run
3) Send code to microcontroller

C++ for Microcontrollers using Arduino

- SO MUCH MORE
  - Classes: C++ is an object oriented programming language, so classes allow you to organize large chunks of code that are repeated
  - Libraries: many libraries exist that add functions such as easier communication with servos, interpreting gyroscope signals, etc

Servo Activity

- What is a Servo Motor?
  - A servo motor is a rotary actuator that allows for precise control of angular position, speed, and acceleration.
  - Commonly used in robotics, RC vehicles, and other applications where precise movement is required.
- Wiring the Servo Motor:
  - 3 Wires:
    - Red Wire: Connect to 5V (power).
    - Brown/Black Wire: Connect to GND (ground).
    - Orange/Yellow Wire: Connect to a PWM-capable digital pin (e.g., pin 9).
- Library for Servo Control:
  - Use the built-in Arduino Servo library for easy control.
  - Provides functions to attach the servo to a pin, set its position, and detach it when done.

```
#include <Servo.h>  // Include the Servo library

Servo myServo;  // Create a servo object

void setup() {
  myServo.attach(9);  // Attach the servo to digital pin 9
}

void loop() {
  myServo.write(0);    // Move the servo to 0 degrees
  delay(1000);         // Wait for 1 second

  myServo.write(90);   // Move the servo to 90 degrees (middle position)
  delay(1000);         // Wait for 1 second

  myServo.write(180);  // Move the servo to 180 degrees
  delay(1000);         // Wait for 1 second
}
```

IEEE UCF

UCF | Colle and

Serial Communication

- What is Serial Communication?
  - Purpose: Allows the Arduino to communicate with a computer or other devices.
  - Use Case: Debugging, sending data to a serial monitor, communicating with other devices (e.g., sensors, modules).
- Why Use Serial Output?
  - Debugging: Print messages to the serial monitor to check variable values, program flow, and detect errors.
  - Data Logging: Send sensor readings or other data to a computer for analysis.
  - Device Communication: Interface with serial peripherals like GPS modules, Bluetooth modules, and other microcontrollers.
- Key Serial Output Functions:
  - Serial.begin(baudRate) Function:
    - Purpose: Initializes serial communication at a specified baud rate (speed).
    - Parameters:
      - baudRate: Speed of communication in bits per second (e.g., 9600).

Serial Communication

- Serial.print(data) Function:
  - Purpose: Sends data to the serial port as human-readable text.
  - Parameters:
    - data: The data to print (can be a string, variable, or expression).
- Serial.println(data) Function:
  - Purpose: Similar to Serial.print(), but adds a newline character after printing.
  - Parameters:
    - data: The data to print (e.g., strings, variables, or expressions).

Serial Communication

- Explanation:
  - Serial.begin(9600): Initializes serial communication.
  - Serial.print(): Prints text to label the output.
  - Serial.println(): Prints the sensor value followed by a newline.
  - Result: Displays continuous sensor readings every second on the serial monitor.
- Key Points:
  - Essential for Debugging: Easily monitor what the microcontroller is doing.
  - Start with Serial.begin(): Always initialize serial communication in setup().
  - Use print vs. println: Choose based on whether you need a new line after printing.
- Serial communication can be used to talk to gyroscopes, gps, bluetooth, and so much more

```
int sensorValue = 0;  // Variable to store sensor reading

void setup() {
  Serial.begin(9600);  // Start serial communication at 9600 baud
}

void loop() {
  sensorValue = analogRead(A0);  // Read sensor value from analog pin A0
  Serial.print("Sensor Value: "); // Print label text
  Serial.println(sensorValue);    // Print the sensor value and move to the
  delay(1000);                    // Wait for 1 second before next reading
}
```

Signup Form

IEEE UCF

Timeline and Important Dates

Weekly Meetings - Estimated Timeline (Starting September 16th)

~~Week 1~~	~~Week 2~~	Week 3	Week 4	Week 5
~~Introduction~~	~~Microcontrollers~~	Fusion 360/Solidworks	KiCAD	Soldering/Prototyping

Next Week's Session:

- CAD (Fusion360/Solidworks) workshop.
- IEEE membership check.

Next session will be on **Monday, September 30**