Week 1 – Course Intro &
First Steps with Arduino

14-Nov
Friday

Embedded AI & Robotics

Who Am I & What This Is

- I’m **Sahil**, Master of AI in Business & your TA
- My world: **AI + Robotics + Embedded Systems**
- **Contact me through whatsapp**
- This course: learn how **data & models** can control **real hardware**
- Very **hands-on**: less talking, more building

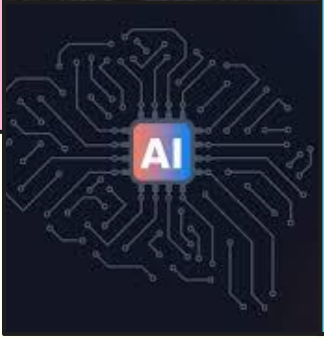
What You’ll Be Able To Do

By The End of This Course You Can Say…

I know what a **microcontroller** is and how use it its potential.

I Have collected data with HW.

I can **wire and program** sensors and motors safely.



I’ve trained and deployed a **AI/ML model** on hardware like a microcontroller.

I can use **Python** to train a robot arm using AI.

I’ve built a **group project** in embedded AI & robotics.

Assessment Overview

Weekly Tutorials

Lab tasks + short reflections
Submit via GitHub / email

Midterm Quiz

Basics of microcontrollers, sensors, motors, AI

Final Group Project

Group of students
Working embedded AI prototype
Live demo + short report

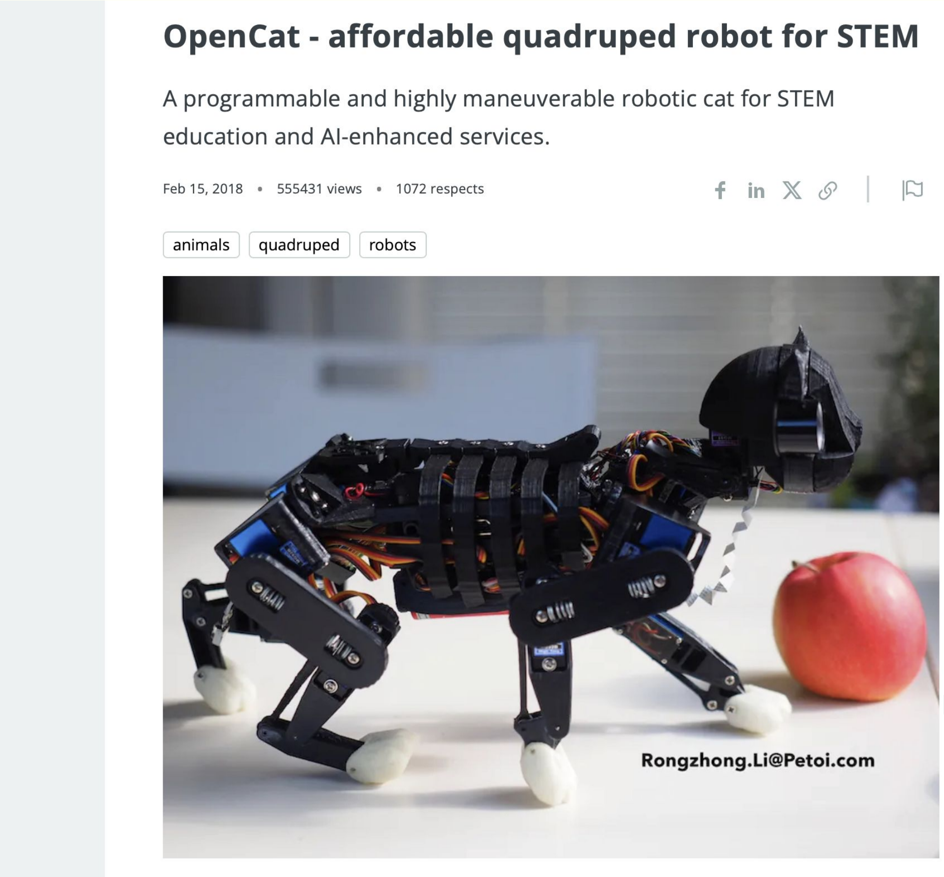
Is This Course for You?

This course *is* for you if:

- You like **learning by building**
- You're curious about collecting DATA using **hardware + AI**
- You want projects that stand out on your **portfolio/LinkedIn**

This course might *not* be for you if:

- You absolutely **hate** touching hardware or wires
- You only want pure theory / coding on a laptop

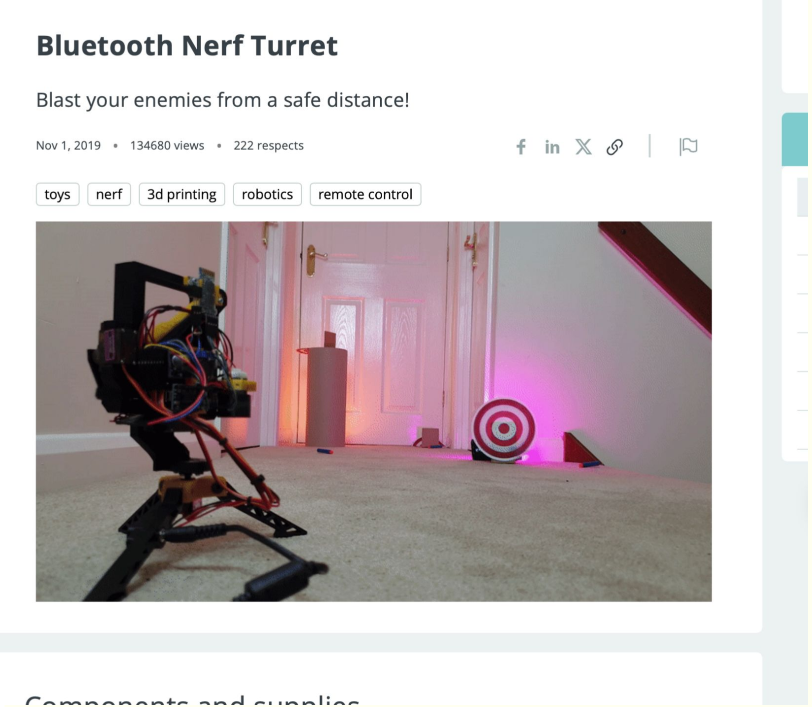


OpenCat - affordable quadruped robot for STEM

A programmable and highly maneuverable robotic cat for STEM education and AI-enhanced services.

Feb 15, 2018 • 555431 views • 1072 respects

animals quadruped robots



Bluetooth Nerf Turret

Blast your enemies from a safe distance!

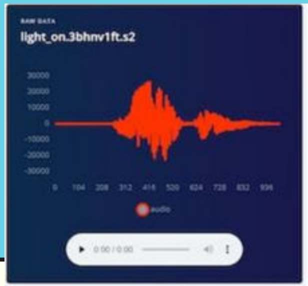
Nov 1, 2019 • 134680 views • 222 respects

toys nerf 3d printing robotics remote control

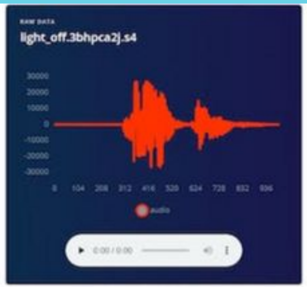
Components and supplies

Projects

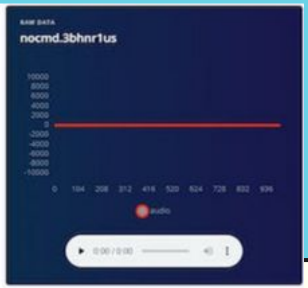
1. On-Device Voice Command Classifier for Smart Homes
2. Gesture-Based Control System with TinyML Classification
3. Edge Computer Vision Security System with Person Detection
4. Predictive Maintenance Dashboard Powered by Sensor Time-Series
5. Emotion-Aware Environment Controller Using Multimodal ML
6. Hyperlocal Weather Forecasting from Self-Collected Sensor Data
7. Smart Irrigation & Crop Health Monitor with ML Recommendations
8. Reinforcement Learning–Driven Autonomous Navigation Robot
9. Biometric Authentication System with Embedded ML on the Edge



light_on



light_off



nocmd

Hardware You’ll Meet

Hardware in This Course

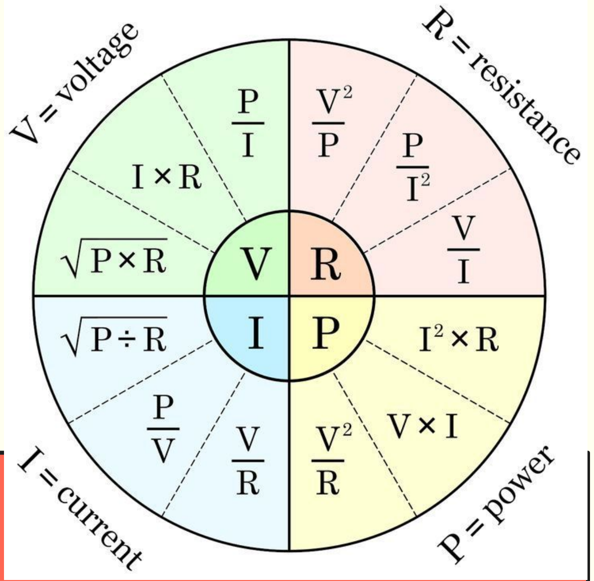
- **Arduino Uno / Nano**
- **ESP32 / ESP32-CAM**
- **Raspberry Pi 5**
- **SO-101 / SO-100 robot arm**
- Sensors:
  - Ultrasonic, LDR, temperature/humidity, accelerometers, etc.
- Motors:
  - DC, servo, stepper

Electricity Basics

Voltage (V): how strong the “push” is (like water pressure)
Current (A): how much is flowing (like water flow)
Resistance (Ω): how much it resists flow (like a narrow pipe)

Why Ground (GND) Matters**

- Ground is the **reference point** (0V)
- All voltages are defined **relative to GND**
- For things to communicate properly:
  - They must **share a common ground**



In our labs:

- **5V** – logic and most sensors/LEDs
- We do **not** play with 220V/240V mains

If GND is not connected:

- Weird / random behaviour
- Sensors and boards don’t agree on “zero”

Overview

What Is a Processor?

- Inside every computer there is a **brain** called a **processor** or **CPU**
- It can:
  - Read instructions (your code)
  - Do calculations
  - Decide what to do next
- It works **very fast**:
  - Millions or billions of steps per second
- On its own, the processor is just a **chip** – it needs other parts to be useful

What Is a Board?

The **processor chip** is hard to use alone
So we put it on a **board** with:

- Power connections (5V, 3.3V, GND)
- Pins connected to the processor
- USB port to talk to a computer
- Extra little parts to keep it stable

Examples of boards:

- Arduino Uno
- ESP32 dev board
- Raspberry Pi (more like a mini-computer board)

Inputs vs Outputs (In Real Life)

- **Input = things the system can *sense***:
  - Buttons, switches
  - Temperature sensor
  - Light sensor (LDR)
  - Microphone, camera, distance sensor
- **Output = things the system can *control***:
  - LEDs, screens
  - Buzzers, speakers
  - Motors (wheels, arms, fans)
  - Relays (turning devices ON/OFF)

Overview

The Basic Loop: Sense → Think → Act

- Almost all embedded systems follow this loop:
    - **Sense** – read inputs
    - **Think** – run logic / model
    - **Act** – control outputs
    - Repeat… **forever**
- Example: automatic room light
    - Sense: light sensor + motion sensor
    - Think: is it dark? is someone here?
    - Act: turn light ON or OFF

Where Does Arduino Fit in the Big Picture?

- Arduino board = **processor + board + pins**
- It sits in the middle:
  Real world → Sensors (input) → Arduino → Outputs (motors, LEDs)
- We connect Arduino to:
    - Our **laptop** (for programming & debugging)
    - The **physical world** (through wires & components)

Why Start with Simple LEDs and Buttons?**

- These examples look **simple**, but they teach:
    - How to connect things safely
    - How to read **inputs**
    - How to control **outputs**
    - How to think in the **“Sense → Think → Act”** loop
- Once you understand this,
    - Swap button → sensor
    - Swap LED → motor
    - Add AI in the “Think” step

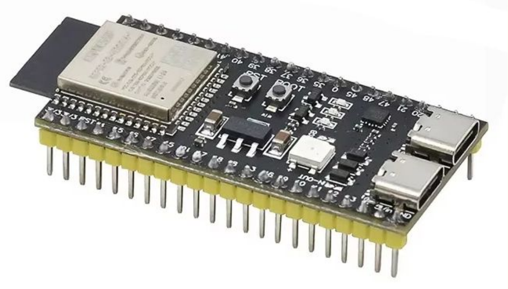
What Is a Microcontroller?

It’s a **tiny computer** on a small chip
It usually runs **one program** that repeats again and again
It has **pins** where you can:

- **Read things** (buttons, sensors) → *inputs*
- **Control things** (LEDs, motors, buzzers) → *outputs*

It has **no screen**, no keyboard, no apps
Very **low power**:

- Can run from a small battery
- Used inside everyday devices (toasters, toys, washing machines)



PC vs Cloud vs Microcontroller

Option 1

Laptop / PC (your Mac or Windows)

- Heavy, powerful machine
- Runs many apps at once (browser, VS Code, Spotify, etc.)
- Great for:
  - Writing and training ML models
  - Running big code, Jupyter notebooks

Option 2

Cloud Server (AWS, Azure, GCP, etc.)

- Like a **giant remote computer** in a data center
- Even more powerful: lots of CPUs, GPUs
- Great for:
  - Training **large models**
  - Handling millions of users
- Needs **internet connection** to reach it

Option 3

Microcontroller (Arduino, ESP32)

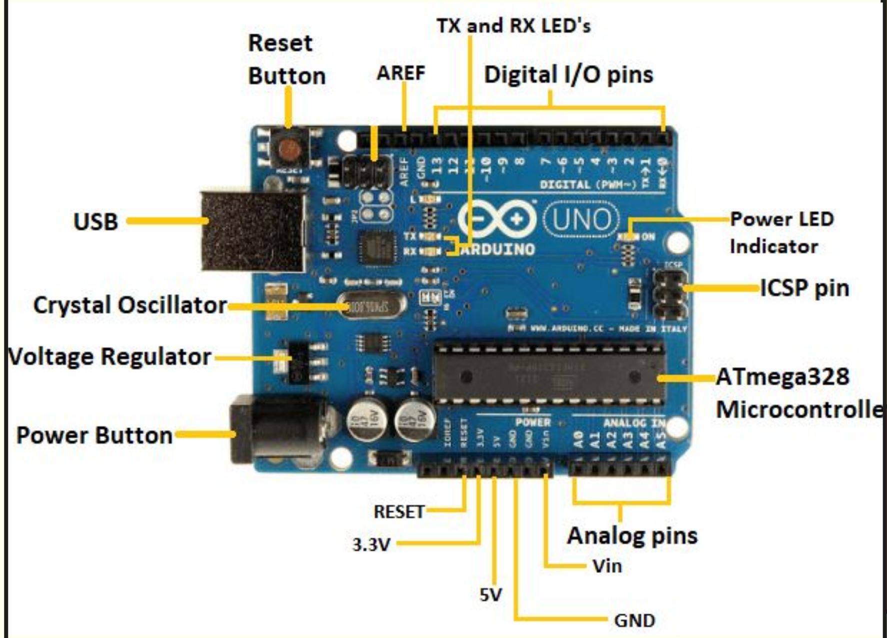
- Very **small and cheap**
- Runs **one program** again and again
- Often has **no operating system**
- Great for:
  - Talking to sensors
  - Controlling motors, lights, buzzers
  - Doing **simple AI** close to the real world
- Can run on a **battery** and sit in a device for years

Why Are We Using Arduino?

Why Are We Using Arduino?

- Arduino is a **family of microcontroller boards**
- Designed to be **beginner-friendly**
  - Clear pin labels: `5V`, `GND`, `13`, `A0` etc.
  - Lots of example projects online
- Big **community**:
  - If you get stuck, someone probably had the same problem
- Simple to connect:
  - Plug in USB → select board → upload code
- Arduino IDE is made for teaching:
  - We write small programs called **“sketches”**
  - Two functions: `setup()` and `loop()`
  - Buttons: **Verify** (check code) and **Upload** (send to board)

The Arduino Board Tour



USB connector (programming + power)

Power jack (we mainly ignore this in lab)

Digital pins (0–13)

Analog pins (A0–A5)

5V & 3.3V pins

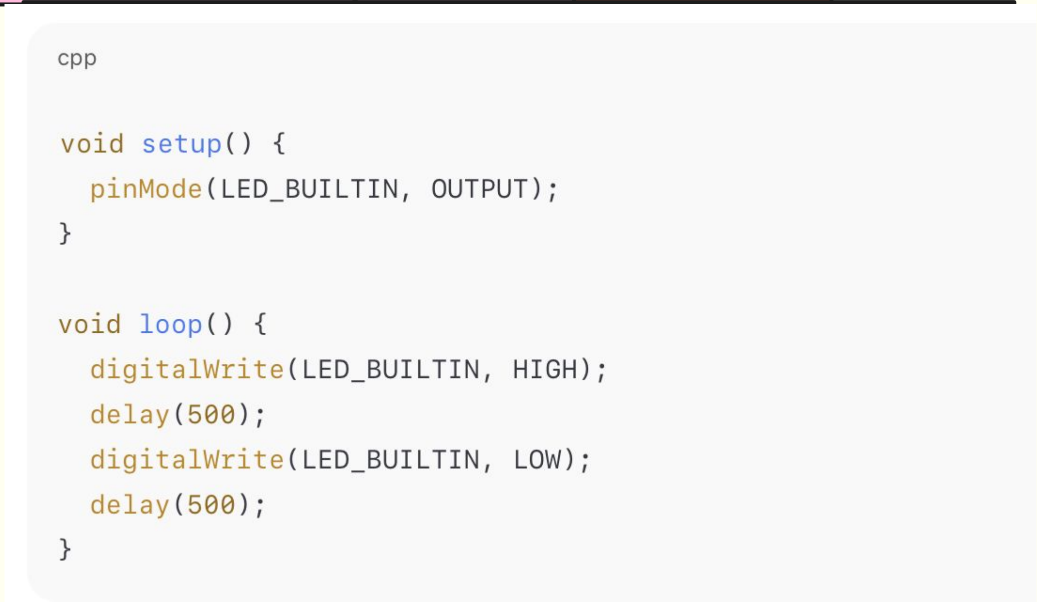
GND pins

Reset button

Power LED & built-in LED (L)

How We Code the Arduino

- Language: **C/C++ (Arduino style)**
- Program = **“sketch”**
- Two main functions:
  - `setup()` → runs **once** at the start
  - `loop()` → runs **forever**
- Typical pattern:
  - `setup()` → configure pins (inputs/outputs)
  - `loop()` → read sensors, decide, control outputs



cpp

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(500);
  digitalWrite(LED_BUILTIN, LOW);
  delay(500);
}
```

Lab Requirements & Expectations

What You Need for Lab

- Bring:
  - Laptop (with permission to install software)
  - Arduino kit (board, USB cable, breadboard, basic components)
- In lab you will:
  - Follow **GitHub step-by-step guides**
  - Complete 2-**3 small tasks** per week
  - Ask lots of questions – confusion is normal

Safety & Lab Rules

Work only with **low-voltage** circuits in this course

Do **not** plug random things into mains power

Double-check wiring **before** powering

If something smells hot, smokes, or is weird:

Disconnect USB / power immediately

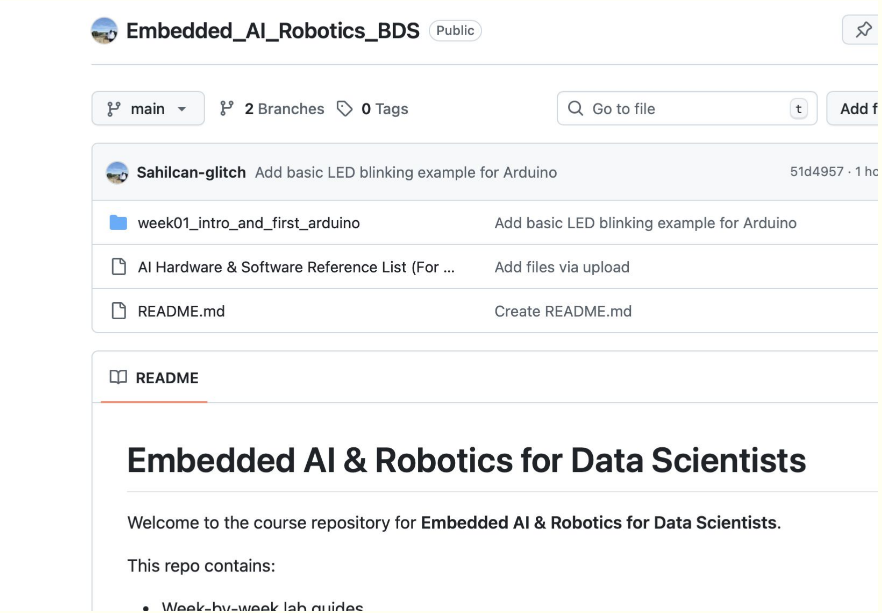
Call me or the instructor

Be kind to the hardware – it's shared

TODAYS TO DO

1. Install **Arduino IDE 2.3.6**
2. Connect Arduino & select **Board + Port**
3. Upload the **Blink** example
4. Try a custom blink pattern
5. Build a **button + LED** circuit
6. (If time) Try two buttons + two LEDs

GitHub Repo for This Course



Sahilcan-glitch Add basic LED blinking example for Arduino 51d4957 · 1 ho

week01_intro_and_first_arduino	Add basic LED blinking example for Arduino
AI Hardware & Software Reference List (For ...	Add files via upload
README.md	Create README.md

README

Embedded AI & Robotics for Data Scientists

Welcome to the course repository for **Embedded AI & Robotics for Data Scientists**.

This repo contains:

- Week-by-week lab guides

All weekly content is here:

- `embedded_ai_robotics_bds` (https://github.com/Sahilcan-glitch/Embedded_AI_Robotics_BDS)

For Week 1:

- `week01_intro_and_first_arduino/`
- `README.md` = step-by-step lab guide
- `INSTALL_AND_SETUP.md` = install instructions with screenshots
- `code_examples/` = example `.ino` files

Q&A

?