tinyCore Complete Setup Guide

From Unboxing to Your First IoT Project

MR.INDUSTRIES

This PDF Sucks for Documentation!

You're viewing a static PDF version with poor formatting and outdated information. **Get the REAL experience with interactive content, the latest versions of code, and better navigation!**

🚀 **View Proper Documentation Website**

Trust us, it's so much better over there! 📱💻✨

Table of Contents

1. Introduction

Welcome to tinyCore!

Congratulations on your new tinyCore! (and Thank You!)

You're well on your way to building some Cool Shit™!

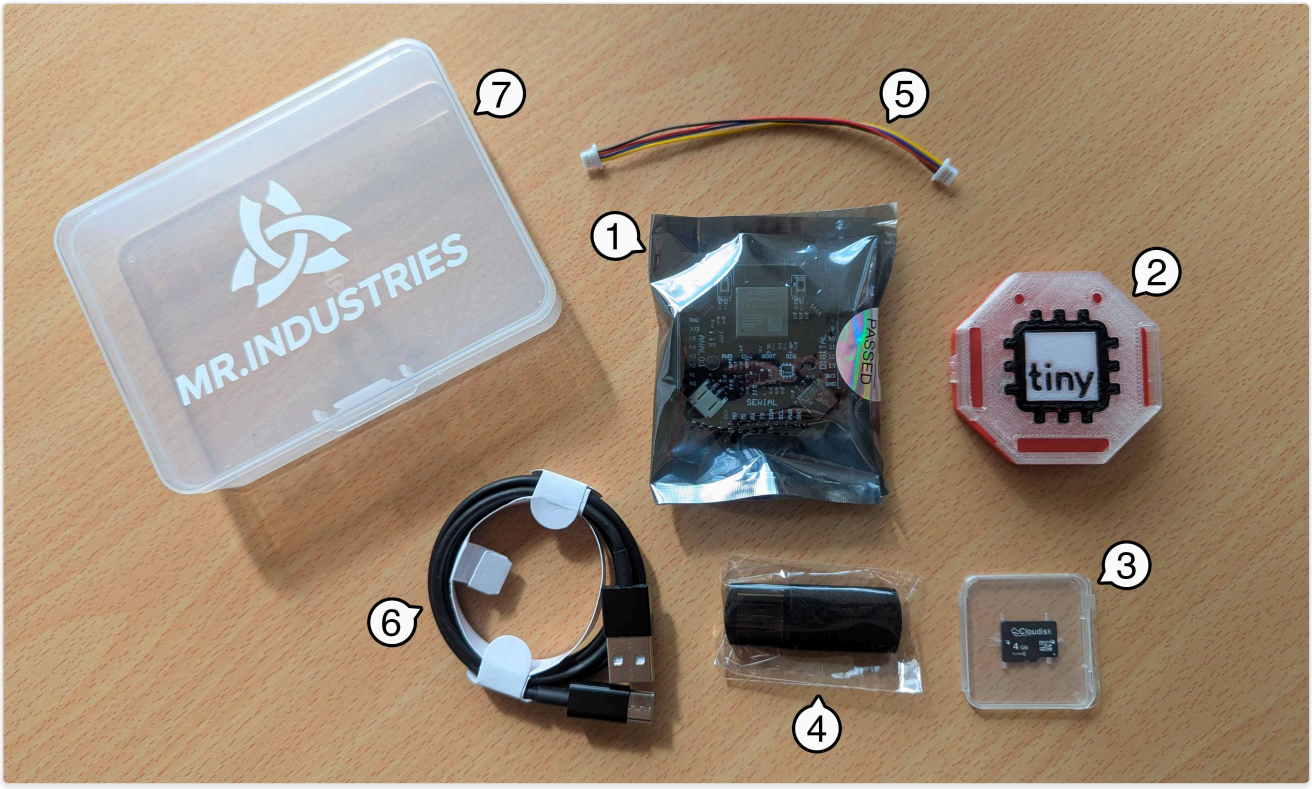
This comprehensive guide will take you from unboxing your tinyCore through building your first complete IoT project. By the end, you'll have:

- A fully assembled tinyCore development board
- A configured development environment
- Hands-on experience with motion sensing
- A working wireless motion tracker that logs data and streams live to your computer

Let's dive in! (And heck, break open those gummy bears!)

2. Unboxing and Assembly

What's Inside Your Kit



Your tinyCore V2 Kit contains everything you need to get started:

Label	Item	Quantity
1	tinyCore V2.0 Development Board	1
2	3D-Printed Multi-color Snap Enclosure	1
NP	M3 Machine Screws (for Enclosure, don't lose these!)	2
3	Micro SD Card (4GB)	1
4	USB Micro SD Card Reader	1
5	STEMMA/QWIIC Cable Connector (100cm)	1
6	USB-C Programming Cable (1m)	1
7	Plastic Project Box (11.5x8.5x2.8cm)	1
NP	Secret sweet treat! 😮	1

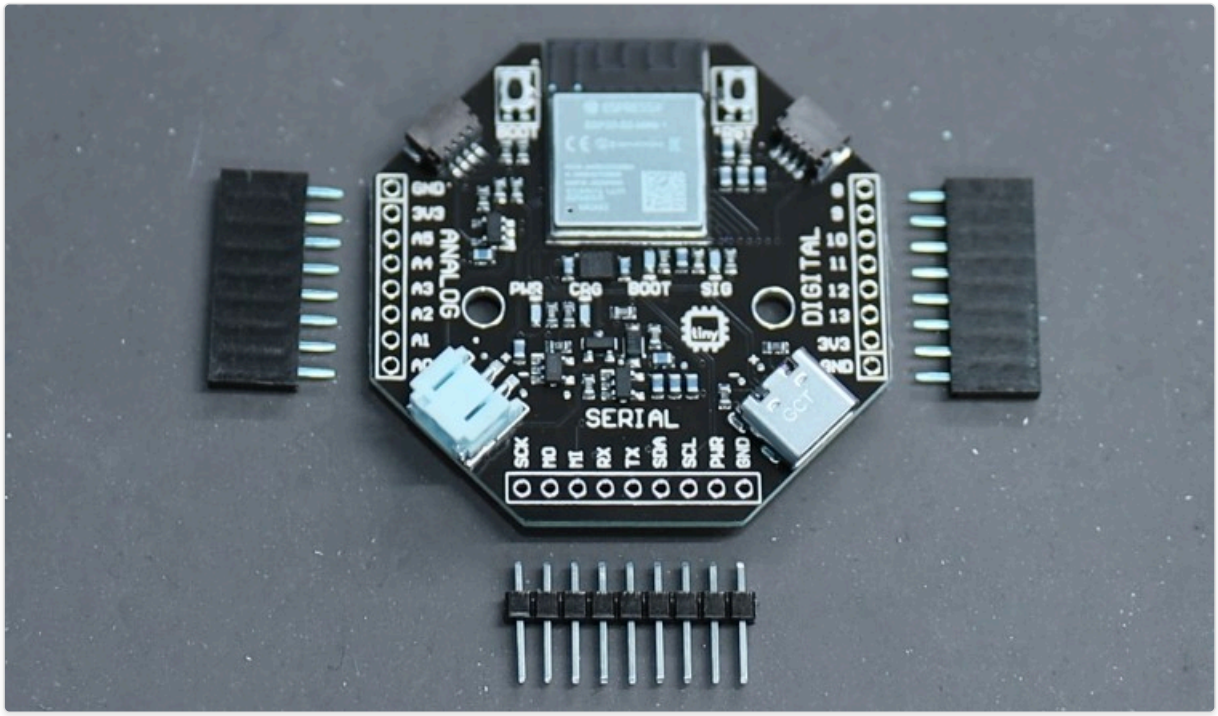
NP = Not Pictured

Preparing the Headers (Optional)

> **Important Note:**
> As of June 2025, if you selected pre-soldered headers during checkout, this step is not necessary!

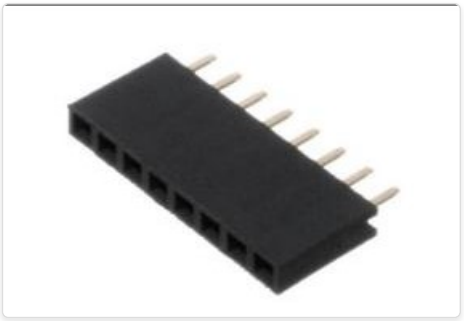
If your headers aren't pre-soldered, you'll need to solder them yourself. This is your first challenge!

You must be careful, since the headers are "keyed" by number of pins. This means they can only be soldered correctly in one configuration.



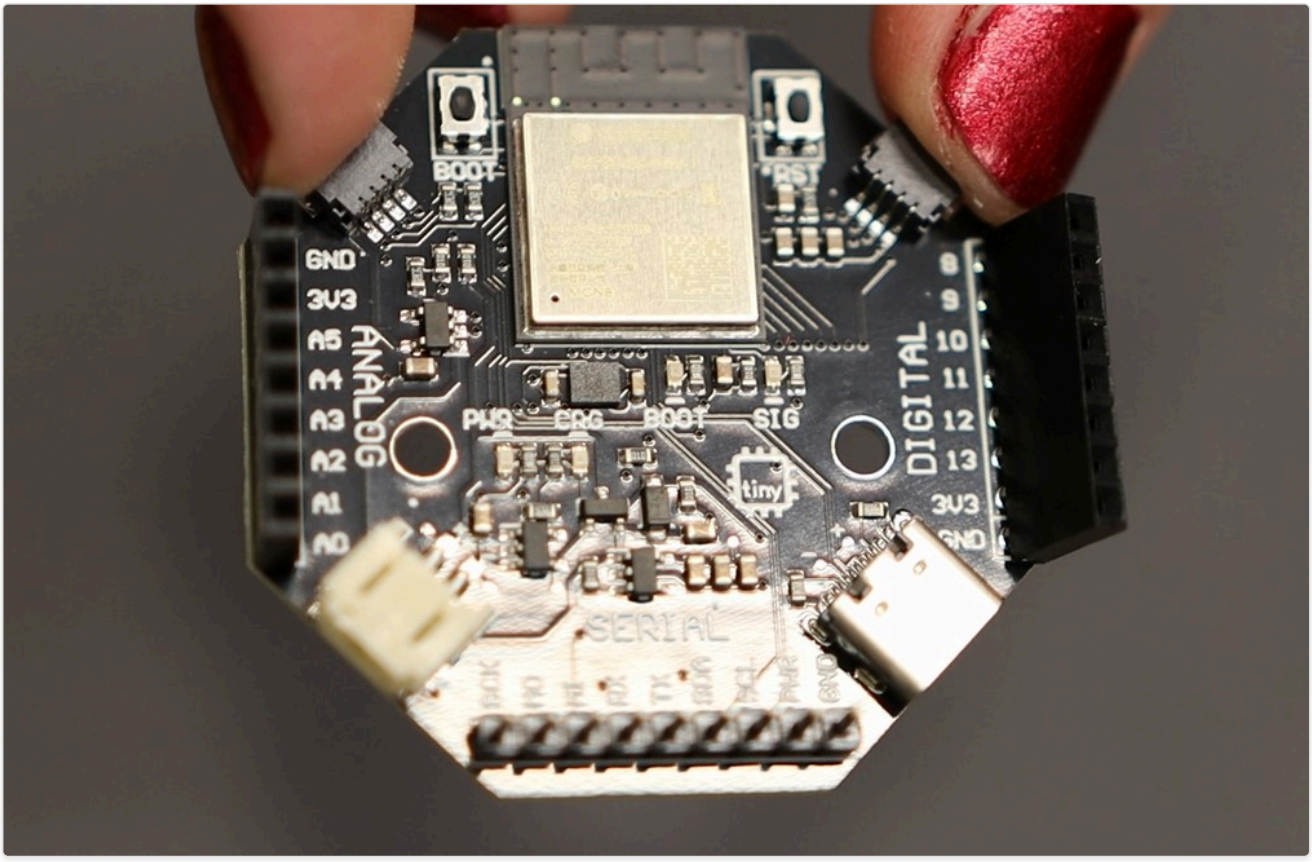
You have three headers to solder:

- Two female headers (8 pins each) - for the left and right (Digital and Analog) pins
- One male header (9 pins) - for the bottom (Serial) pins



Soldering Instructions:

1. Place the female headers while the PCB is face up
2. Flip the board over and solder the female headers
3. Place the male header pins (short end through the board)
4. Solder the male headers

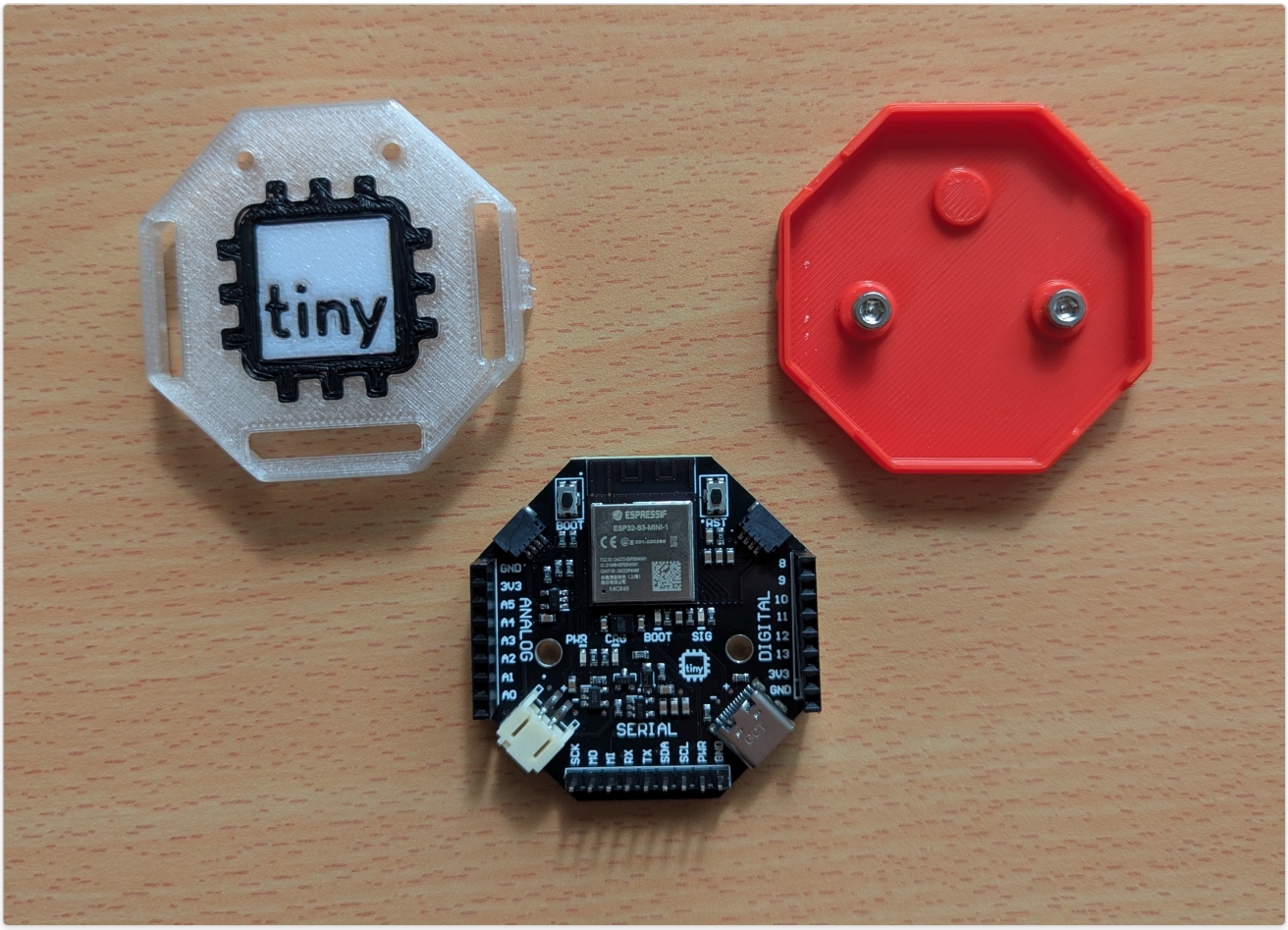


> **Critical:**
> Make sure you put the short end of the male headers through the board!

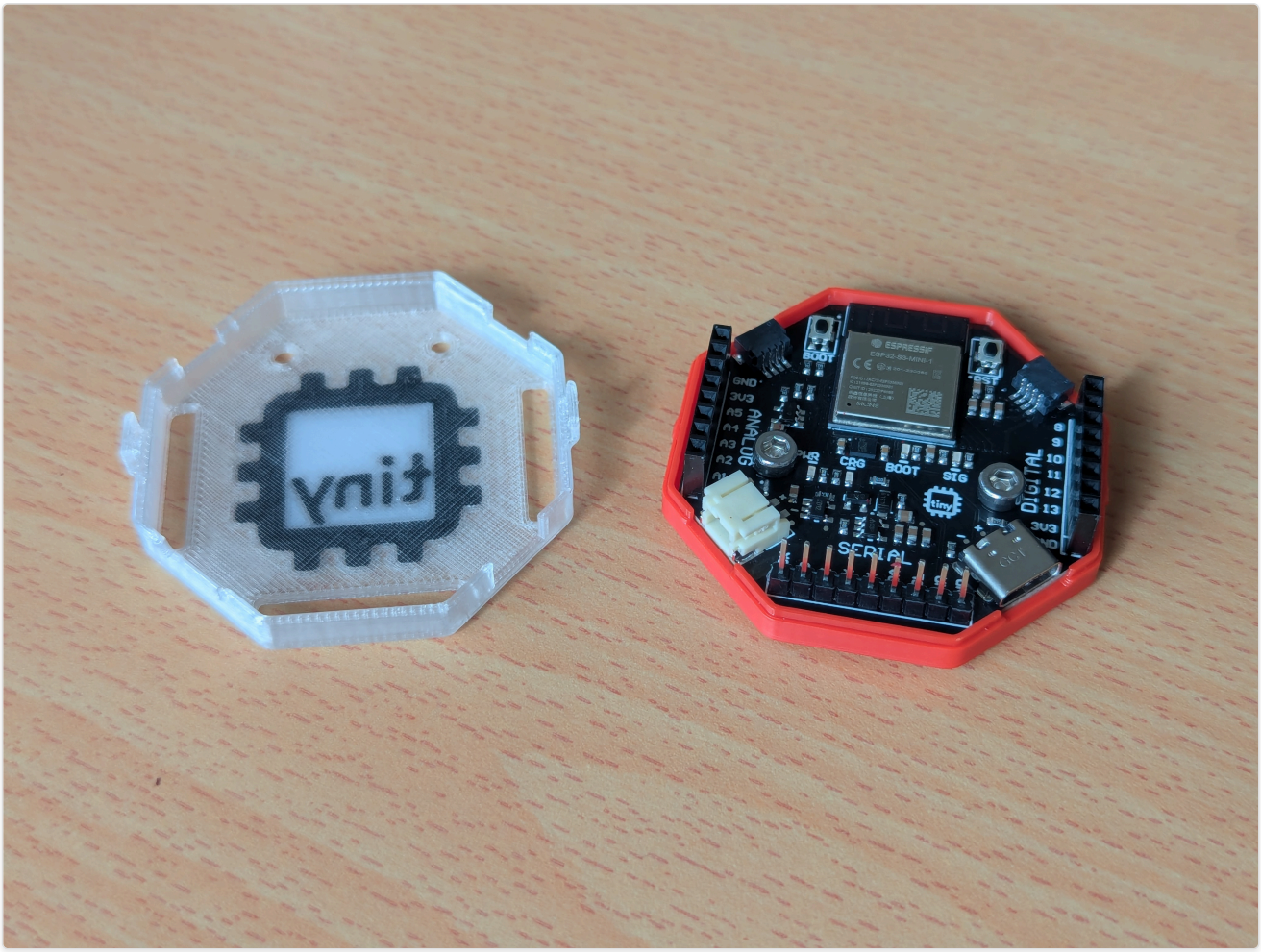
Assembling the Enclosure

Once the headers are ready, let's put the PCB in its protective enclosure:

1. **Open the enclosure:** Gently unclip the hinges and open the clamshell body.

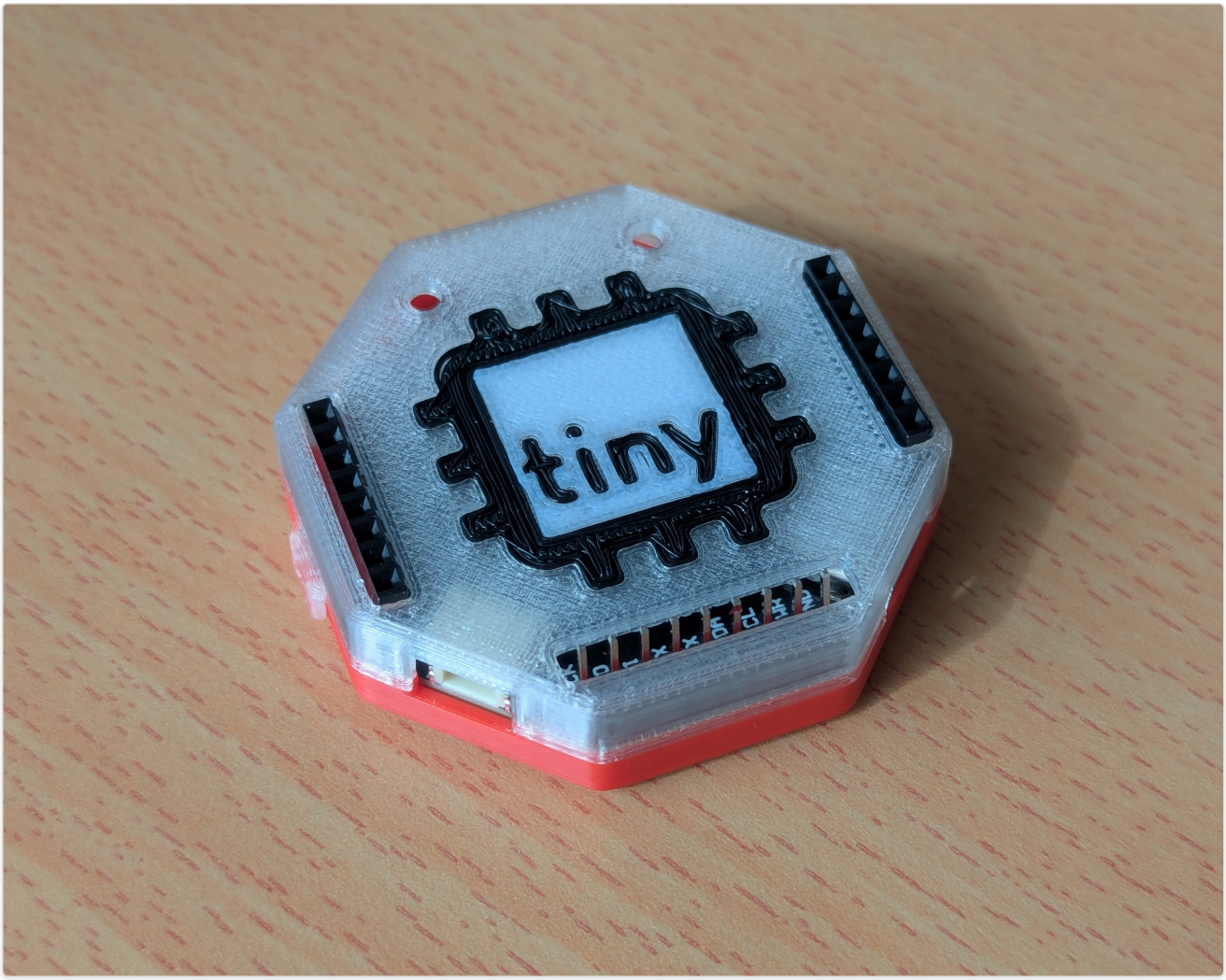


2. **Position the PCB:** Remove the screws and place the PCB so the screw holes align. The solid plastic dot goes towards the top!
3. **Secure with screws:** Replace and tighten the M3 screws (snug, not overtight).



> **Do not overtighten the screws!**
>
> Just make sure they are snug to avoid damaging the enclosure.

4. **Close the enclosure:** Return the cover to its original position, ensuring the headers align with the holes in the top.



Your tinyCore is now ready for programming!

3. Arduino IDE Setup

Now we'll set up your development environment to program the tinyCore. This process installs the necessary board definitions and tools.

Prerequisites

- **Arduino IDE 1.6.4 or later** (we recommend the latest version)
- Working internet connection

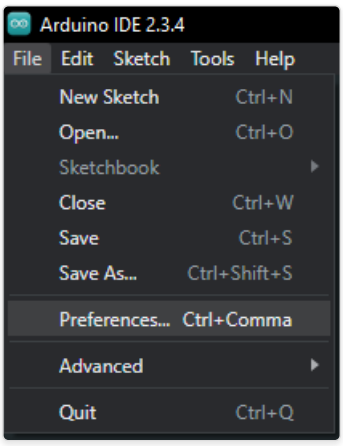
Installation Steps

1. Download and Install Arduino IDE

1. Download the Arduino IDE from the Arduino Software page
2. Install the Arduino IDE on your local machine
3. Open the Arduino IDE

2. Access Preferences

1. Navigate to Preferences:
   - **Windows/Linux:** File > Preferences
   - **macOS:** Arduino > Preferences

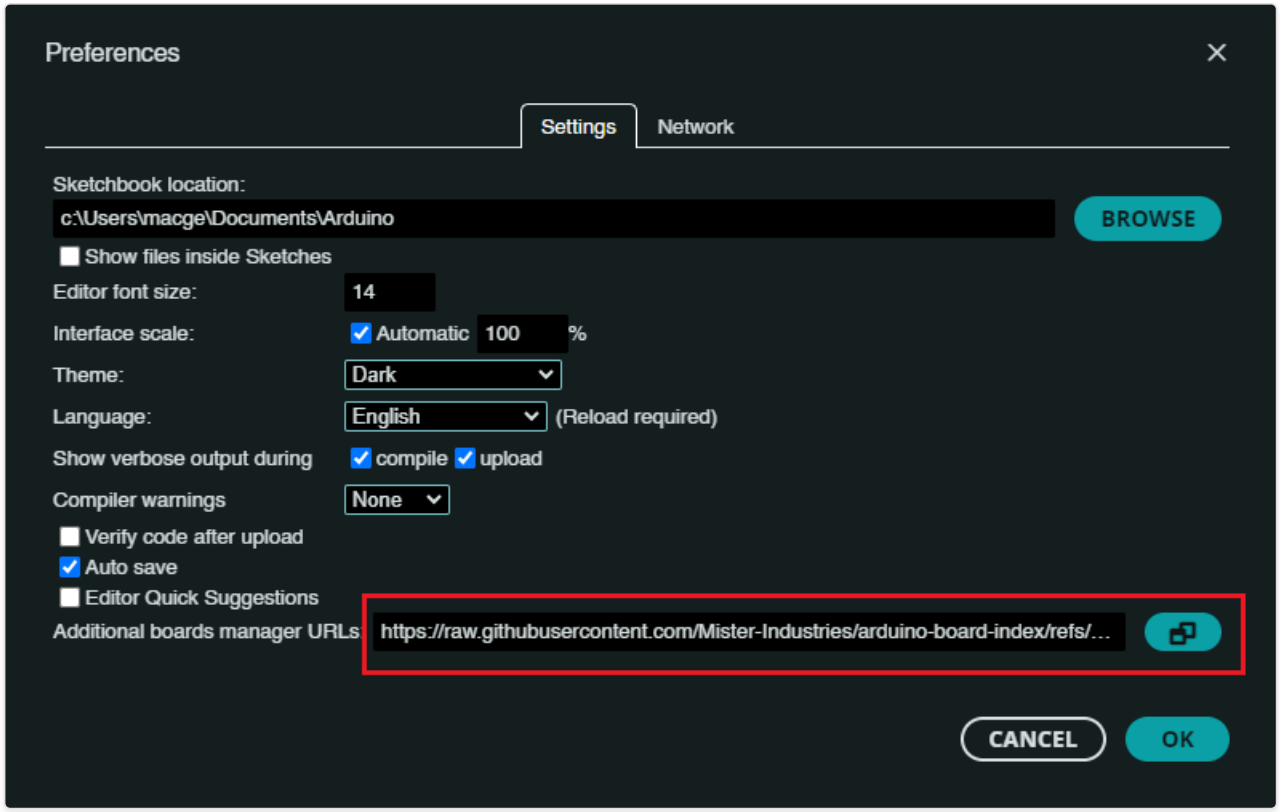


3. Add Custom Board Manager URL

1. In the "Additional Board Manager URLs" field, paste:

```
https://raw.githubusercontent.com/Mister-Industries/arduino-board-index/refs/heads/main/package_tiny_core_index.json
```

For multiple URLs, separate with commas.

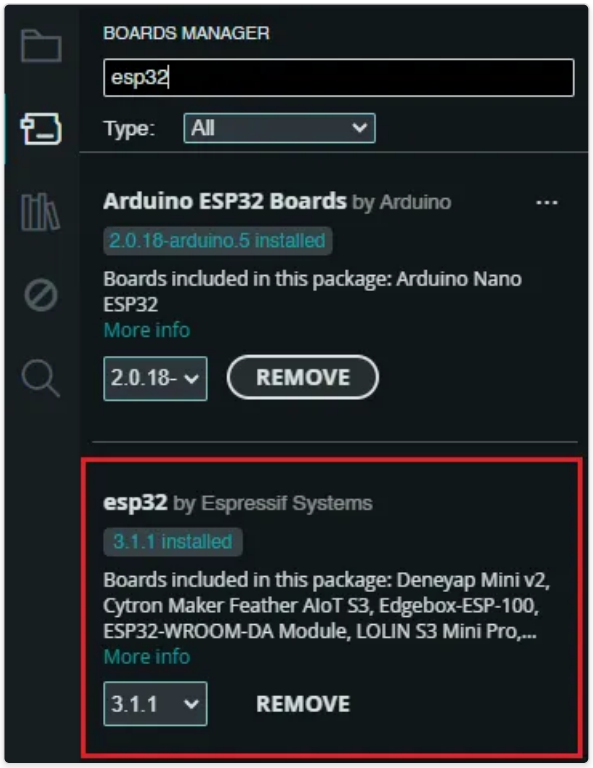


4. Open Board Manager

1. Go to Tools > Board > Boards Manager
2. Or press the Boards Manager icon on the left menu

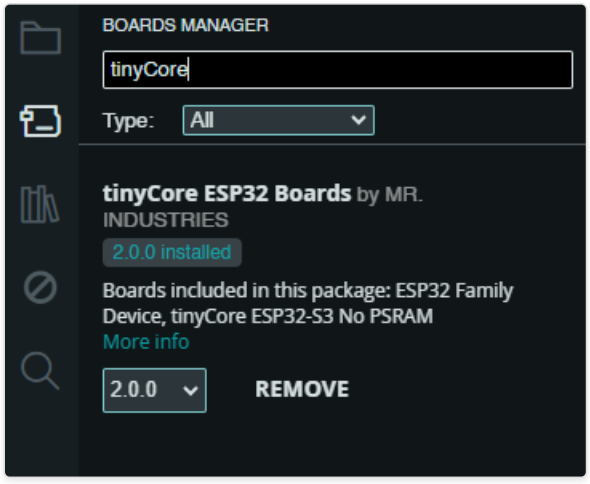
5. Install ESP32 Boards/Tools

1. Search for `esp32`
2. Select `esp32` package (v3.1.1) by Espressif
3. Click "Install"



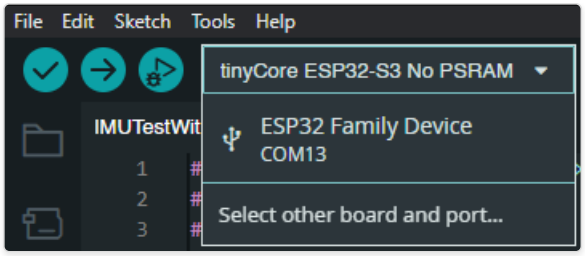
6. Install tinyCore Custom Board

1. Search for `tinyCore`
2. Select `tinyCore ESP32 Boards` package (v2.0.0) by MR.INDUSTRIES
3. Click "Install"



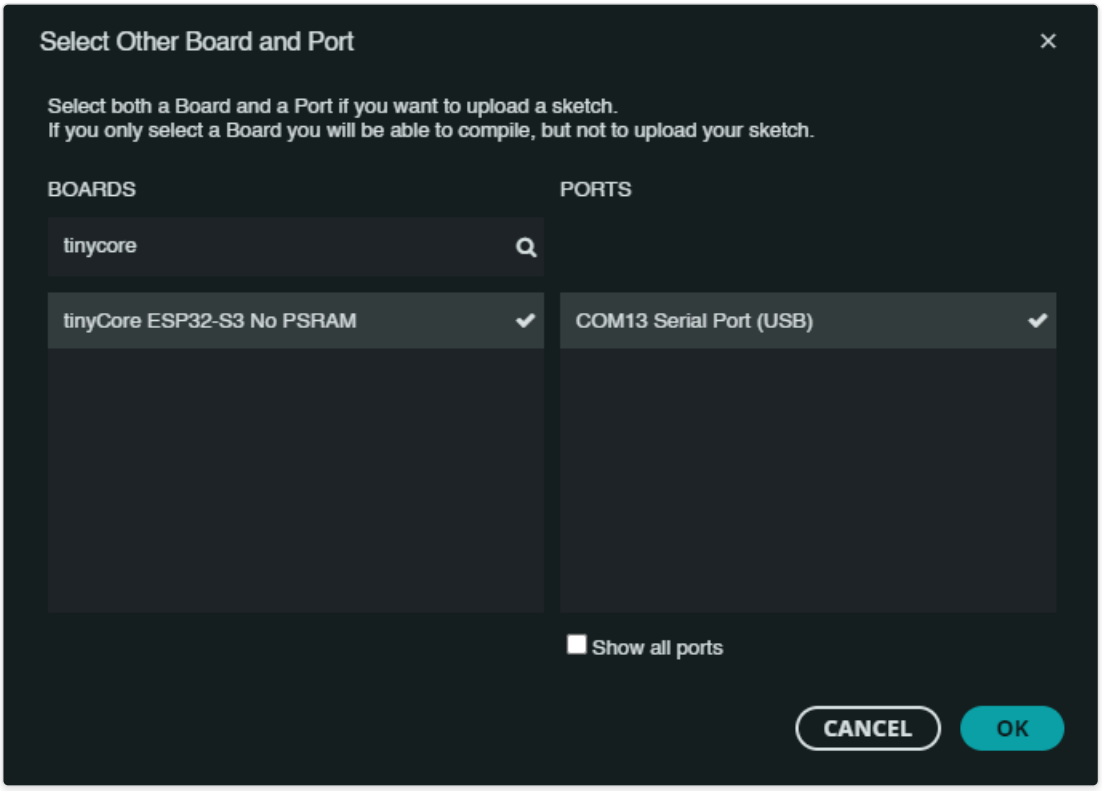
7. Connect Your tinyCore

1. Plug in your tinyCore using the included USB-C cable
2. Under the devices tab, you should see a new device appear
3. Click `Select other board and port...`



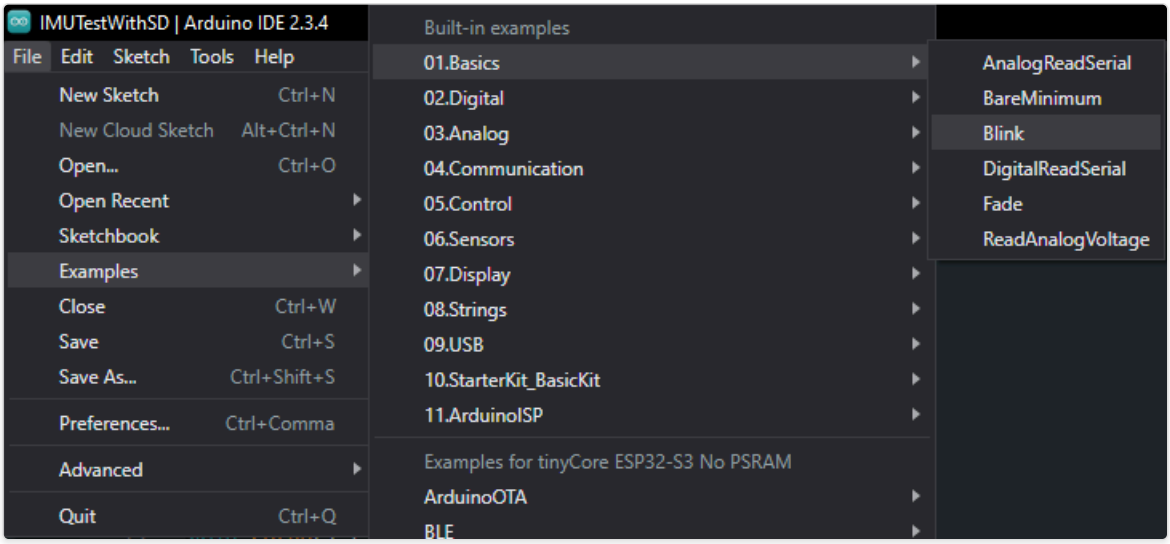
8. Select Board and Port

1. Search "tinyCore" and select `tinyCore ESP32-S3 No PSRAM`
2. Click on the COM Port of the new device
3. Click OK



9. Test with Blink Example

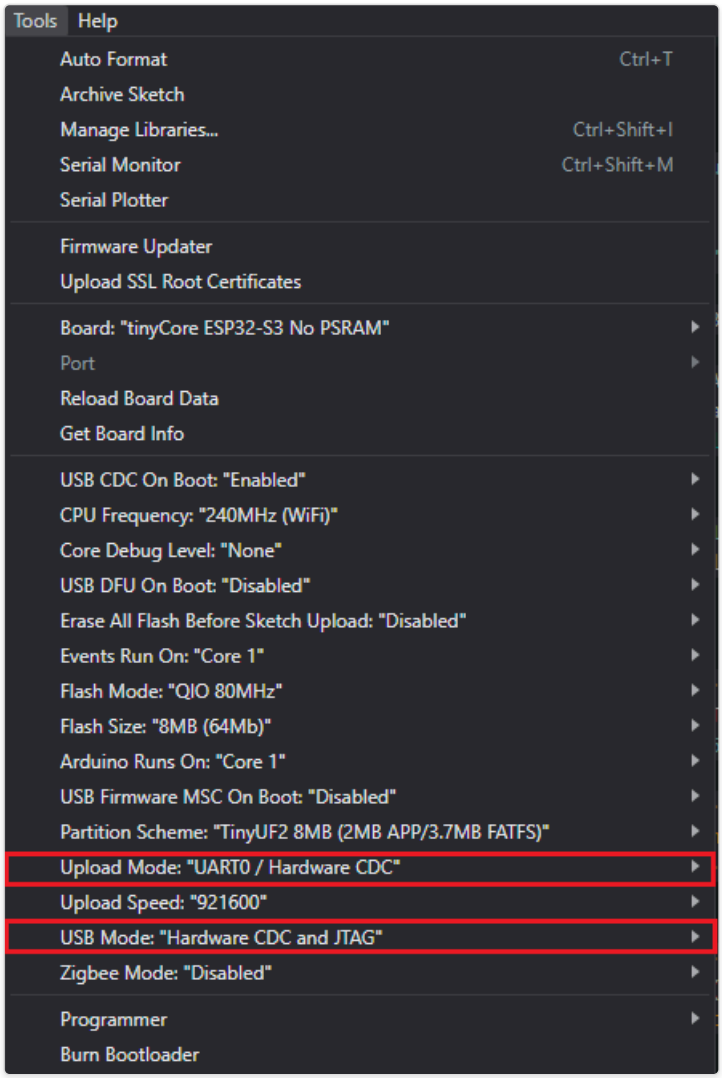
1. Open the Blink example: `File → Examples → 01.Basics → Blink`



10. Configure Upload Settings

Under `Tools`, make sure these settings are correct:

- Upload Mode: `UART0 / Hardware CDC`
- USB Mode: `Hardware CDC and JTAG`



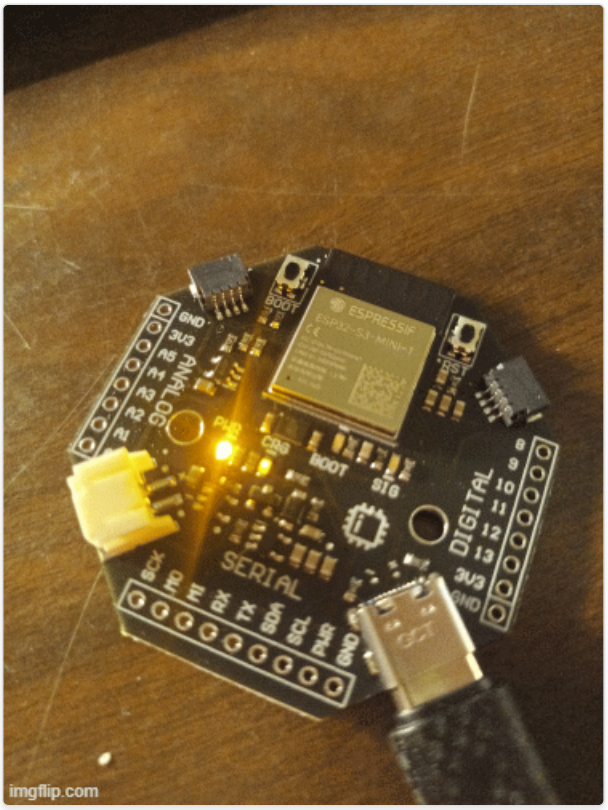
11. Upload and Test

1. Click Upload!
2. The code should compile and you should see the device flash and hard reset
3. The "SIG" LED should now be blinking once every second!

```
Writing at 0x00010000... (10 %)
Writing at 0x0001bc56... (20 %)
Writing at 0x00027d4c... (30 %)
Writing at 0x0002d4bf... (40 %)
Writing at 0x000333f9... (50 %)
Writing at 0x00038d5a... (60 %)
Writing at 0x0003e624... (70 %)
Writing at 0x00043f54... (80 %)
Writing at 0x0004b64d... (90 %)
Writing at 0x00054b27... (100 %)
Wrote 296688 bytes (157500 compressed) at 0x00010000 in 1.9 seconds (effective 1268.2 kbit/s)...
Hash of data verified.
Compressed 196208 bytes to 126577...
Writing at 0x00410000... (12 %)
Writing at 0x004180e8... (25 %)
Writing at 0x00420356... (37 %)
Writing at 0x00425c6b... (50 %)
Writing at 0x0042b35a... (62 %)
Writing at 0x00430c58... (75 %)
Writing at 0x00435e29... (87 %)
Writing at 0x0043b82c... (100 %)
Wrote 196208 bytes (126577 compressed) at 0x00410000 in 1.3 seconds (effective 1232.4 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting with RTC WDT...
```

Uploading...

CANCEL



Success!

Congratulations! You're now ready to start building with your tinyCore.

4. Understanding and Using the IMU

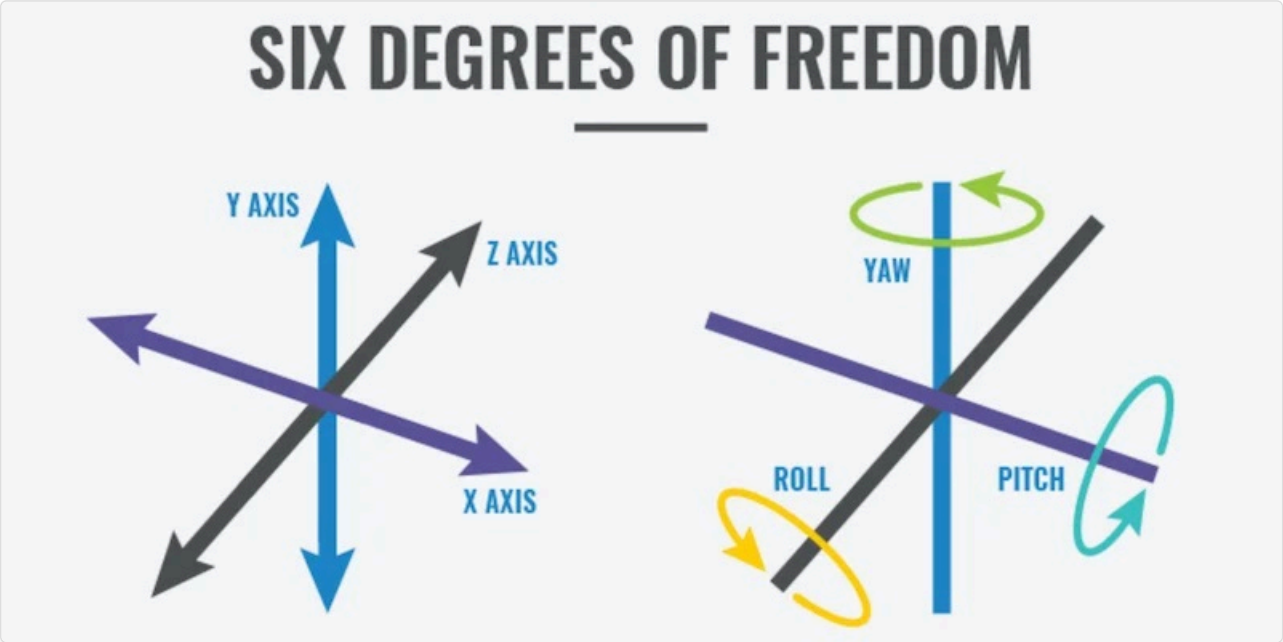
> **Prerequisites Check:**
>
> Make sure you have completed the Arduino IDE setup from the previous section before continuing!

What is an IMU?

An IMU stands for "Inertial Measurement Unit", but basically, it's a motion tracker! The tinyCore's IMU has **6 "Degrees of Freedom"** (DOF), meaning we measure 6 things:

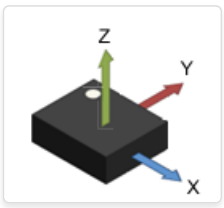
- **Acceleration** in the X, Y, and Z axes
- **Rotation** (gyroscopic motion) in the X, Y, and Z axes

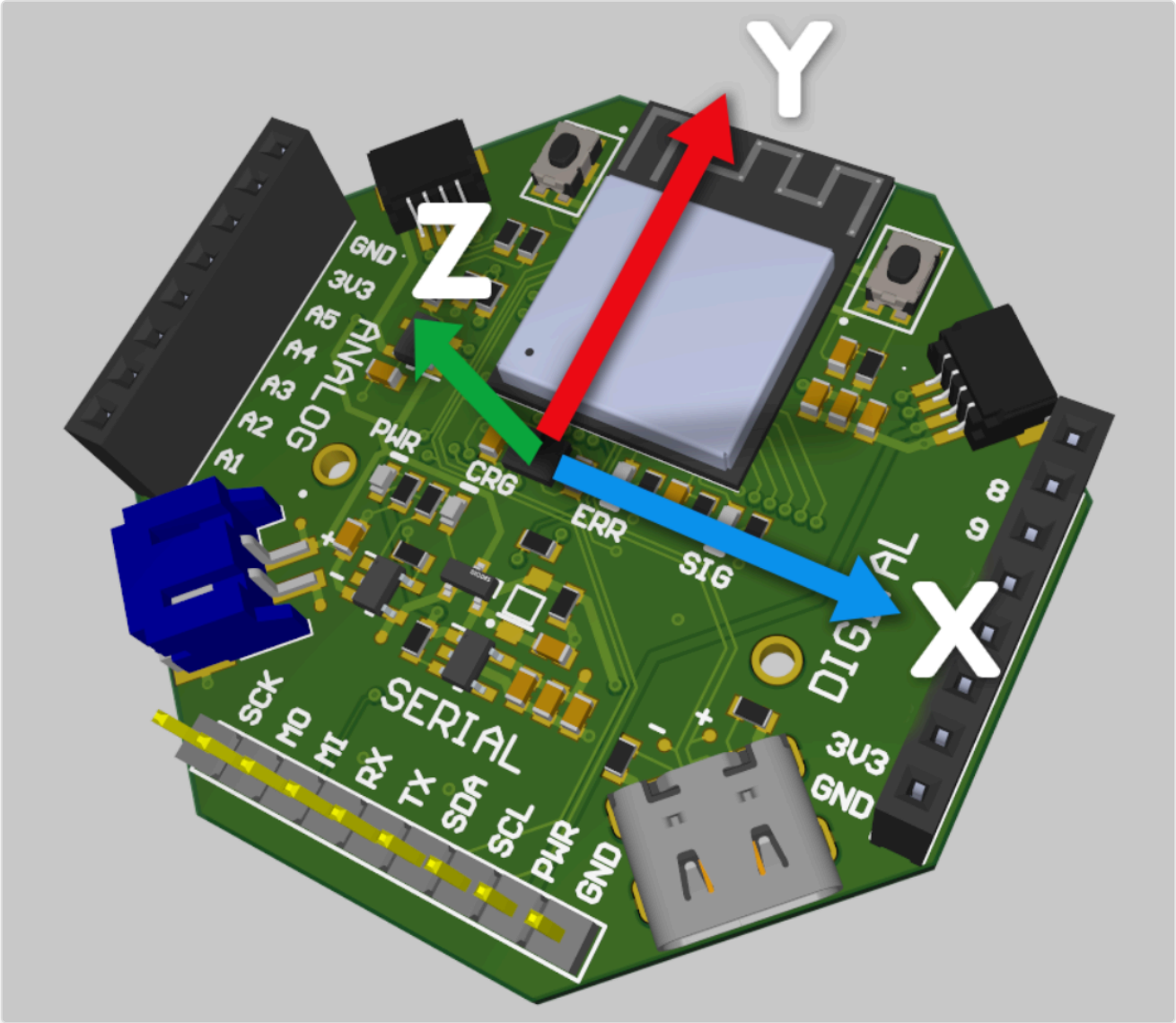
This is also known as Pitch, Yaw, and Roll for aerospace applications.



IMU Hardware Details

The tinyCore uses the LSM6DSOX IMU manufactured by STMicroelectronics. Here's how it's oriented:





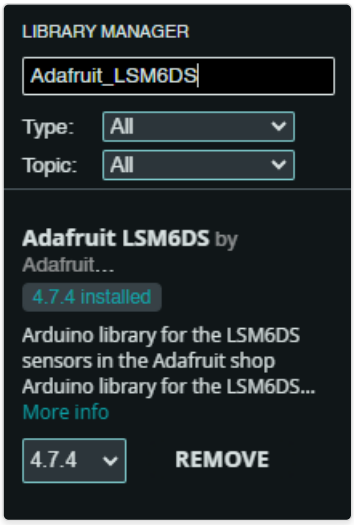
Setting Up the IMU Library

1. Install Required Libraries

The LSM6DSOX has an excellent Arduino library made by Adafruit. Install it using the Library Manager:

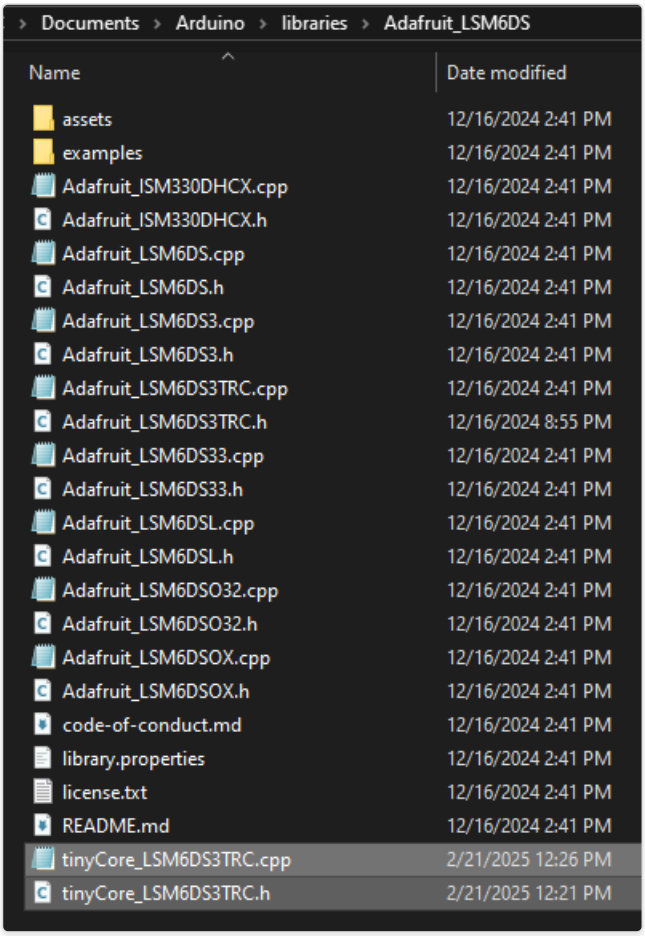
1. Click the Library Manager icon in Arduino IDE

2. Search "Adafruit_LSM6DS" and click `INSTALL`



For Legacy (iotaCore) Boards Only:

Legacy boards (labeled "iotaCore" on the back) use an older LSM6DS3TRC IMU chip with a different I2C address (0x69 instead of 0x6A). If you have a legacy board, you'll need our custom library files. After installing the Adafruit library, navigate to `Documents → Arduino → libraries → Adafruit LSM6DS` and add the custom files: tinyCore_LSM6DS3TRC.cpp and tinyCore_LSM6DS3TRC.h



IMU Example Code

This example demonstrates how to initialize the IMU and view its data using Arduino's Serial Plotter. Copy this code into a new Arduino sketch:

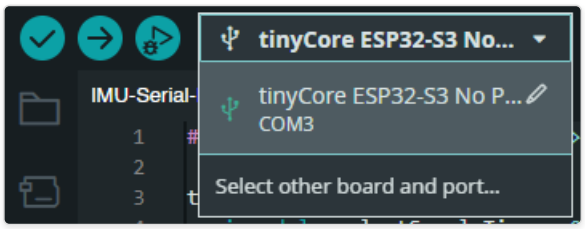
```
#include <Adafruit_LSM6DSOX.h>

Adafruit_LSM6DSOX lsm6dsox;
unsigned long lastSampleTime = 0;
const unsigned long SAMPLE_INTERVAL = 25; // Sample every 25ms

void setup() {
  Serial.begin(115200);
  /* while (!Serial) {
    delay(10);
  }*/

  // Initialize IMU-related pins
  pinMode(6, OUTPUT);
  digitalWrite(6, HIGH);

  // Initialize I2C
  Wire.begin(3, 4);
  delay(100);

  Serial.println("Scanning for I2C devices...");
  for (byte address = 1; address < 127; address++) {
    Wire.beginTransmission(address);
    byte error = Wire.endTransmission();
    if (error == 0) {
      Serial.print("I2C device found at address 0x");
      if (address < 16) {
        Serial.print("0");
      }
      Serial.println(address, HEX);

      // If this is our LSM6DSOX address, try reading WHO_AM_I register
      if (address == 0x6A || address == 0x6B) {
        Wire.beginTransmission(address);
        Wire.write(0x0F);  // WHO_AM_I register address
        Wire.endTransmission(false);
        Wire.requestFrom(address, 1);
        if (Wire.available()) {
          byte whoAmI = Wire.read();
          Serial.print("WHO_AM_I register value: 0x");
          Serial.println(whoAmI, HEX);
          // Should be 0x6C for LSM6DSOX
        }
      }
    }
  }

  Serial.println("Attempting to initialize LSM6DSOX...");
  if (!lsm6dsox.begin_I2C()) {
    Serial.println("Failed to find LSM6DSOX chip");
    Serial.println("Check your wiring!");
    while (1) {
      delay(10);
    }
  }

  Serial.println("LSM6DSOX Found!");

  // Configure IMU settings
  lsm6dsox.setAccelRange(LSM6DS_ACCEL_RANGE_2_G);
  lsm6dsox.setGyroRange(LSM6DS_GYRO_RANGE_250_DPS);
  lsm6dsox.setAccelDataRate(LSM6DS_RATE_104_HZ);
  lsm6dsox.setGyroDataRate(LSM6DS_RATE_104_HZ);

  // Print labels for Serial Plotter
  Serial.println("AccelX:AccelY:AccelZ:GyroX:GyroY:GyroZ:Temp");
```

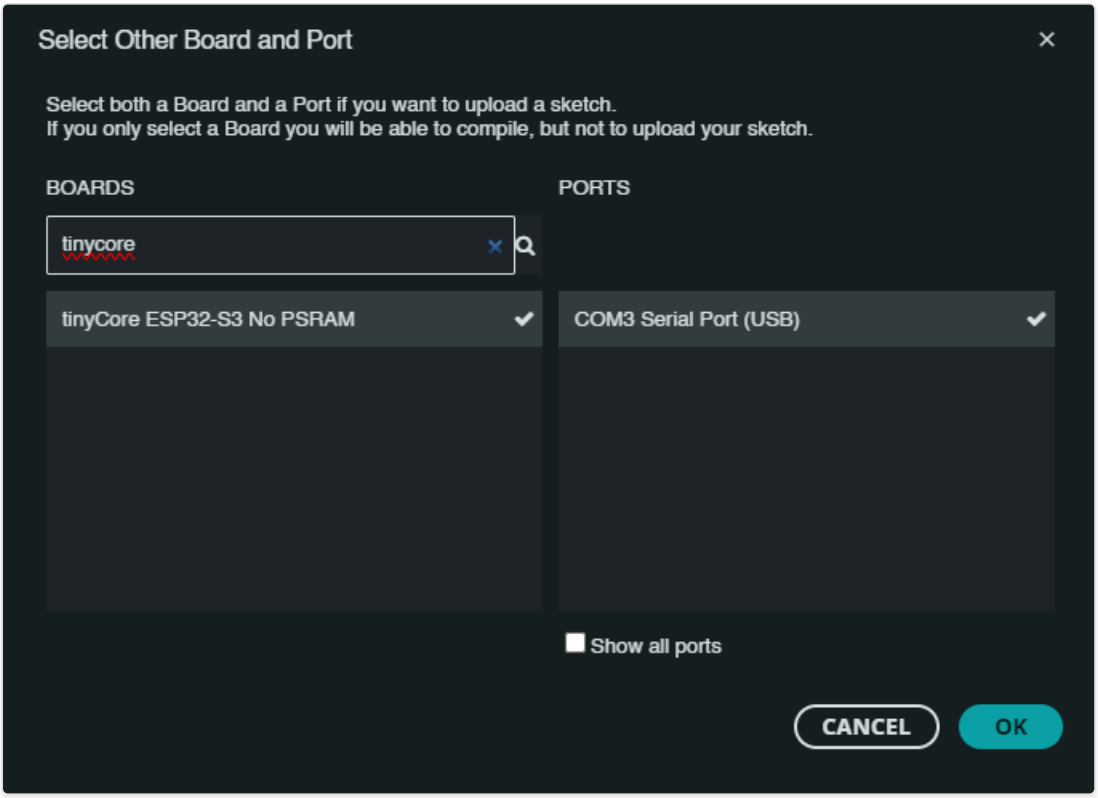
```
}

void sampleIMUData() {
  sensors_event_t accel;
  sensors_event_t gyro;
  sensors_event_t temp;
  lsm6dsox.getEvent(&accel, &gyro, &temp);

  // Format data for serial plotter (label:value:label:value format)
  Serial.print("AccelX:");
  Serial.print(accel.acceleration.x);
  Serial.print(",");
  Serial.print("AccelY:");
  Serial.print(accel.acceleration.y);
  Serial.print(",");
  Serial.print("AccelZ:");
  Serial.print(accel.acceleration.z);
  Serial.print(",");
  Serial.print("GyroX:");
  Serial.print(gyro.gyro.x);
  Serial.print(",");
  Serial.print("GyroY:");
  Serial.print(gyro.gyro.y);
  Serial.print(",");
  Serial.print("GyroZ:");
  Serial.print(gyro.gyro.z);
  Serial.print(",");
  Serial.println(temp.temperature);
}

void loop() {
  unsigned long currentTime = millis();

  // Sample data at specified interval
  if (currentTime - lastSampleTime >= SAMPLE_INTERVAL) {
    sampleIMUData();
    lastSampleTime = currentTime;
  }
}
```

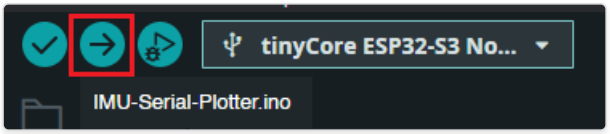
Testing the IMU

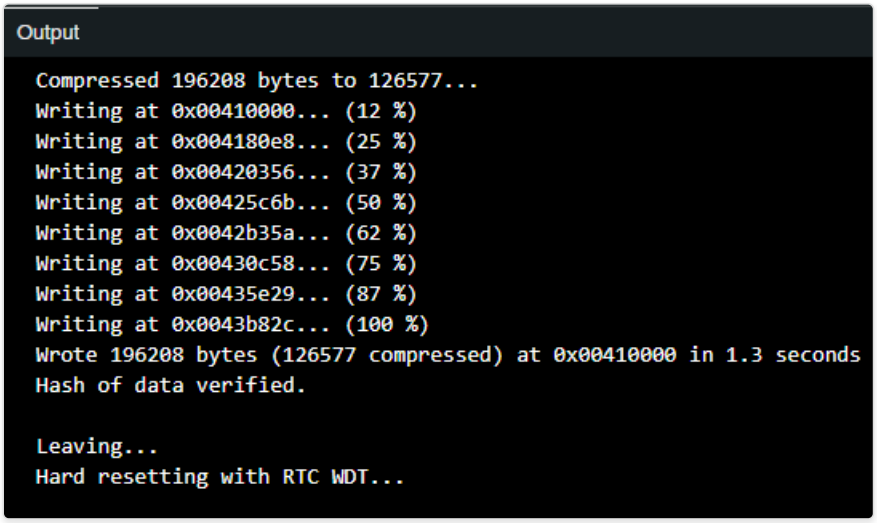
1. Upload the Code

1. Select your tinyCore board and COM port
2. Upload the code and wait for the board to reboot



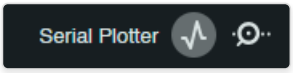






2. Open the Serial Plotter

1. Click the Serial Plotter icon in the top right corner
2. Ensure baud rate is set to 115200



3. Observe the Data

You should see several color-coded lines representing different sensor readings:

If your tinyCore is sitting upright on a flat surface, you should see the AccelZ line hovering around 9.81 m/s^2 - this is gravity!

4. Experiment with Motion

Try these movements and observe the patterns:

- **Shake the device:** Watch acceleration spikes in all directions
- **Draw circles in the air:** Look for sine wave patterns
- **Try different orientations:** See how gravity affects different axes


COM3
AccelX
AccelY
AccelZ
GyroX
GyroY
GyroZ
Interpolate
STOP
20
15
10
5
0
-5
-10
8836
8848
8860
8872
8885
Type Message
SEND
New Line
115200 baud

Brain Teaser:

In the circular motion example above, if we're sampling every 25ms and there are ~13 points per rotation, we can calculate the frequency: 25 × 13 = 325ms per cycle. Converting to frequency: 1/0.325 ≈ 3Hz!

5. Building Your First Project: Motion Tracker

Now it's time to combine everything you've learned into a complete IoT project! You'll build a smart motion tracker that can log data locally and stream it wirelessly to your computer for real-time visualization.

What We're Building

This motion tracker combines all the special features of the tinyCore:

- **Motion Sensing:** Using the built-in IMU
- **Local Data Storage:** Logging to the SD card
- **Wireless Data Streaming:** Real-time transmission via WiFi

Real-World Applications

Students have used this exact setup to create:

- Fitness rep counters and form analyzers
- Musical gesture controllers
- Physics lab acceleration experiments
- Door entry monitoring systems
- Even a "FitBit for cows" (seriously!)

Complete Motion Tracker Code

Copy this comprehensive code into a new Arduino sketch:

```
#include <Adafruit_LSM6DSOX.h>
#include <WiFi.h>
#include <WiFiUdp.h>
#include <SD.h>
#include <SPI.h>

// WiFi credentials - Change these!
const char* ssid = "YOUR_WIFI_NAME";
const char* password = "YOUR_WIFI_PASSWORD";
const char* host_ip = "192.168.1.100"; // Your computer's IP
const int udp_port = 12345;

// Hardware setup
Adafruit_LSM6DSOX lsm6dsox;
WiFiUDP udp;
File dataFile;

// Timing and data
unsigned long lastSample = 0;
const unsigned long SAMPLE_INTERVAL = 50; // 20Hz sampling
unsigned long sessionStart;
int packetCount = 0;

// Motion data structure
struct MotionData {
  float accelX, accelY, accelZ;
  float gyroX, gyroY, gyroZ;
  float temp;
  unsigned long timestamp;
};

void setup() {
  Serial.begin(115200);
  delay(1000);

  Serial.println("=== tinyCore Motion Tracker Starting ===");

  // Initialize IMU
  setupIMU();

  // Initialize SD Card
  setupSDCard();

  // Connect to WiFi
  setupWiFi();

  sessionStart = millis();
  Serial.println("Motion tracker ready! Starting data collection...");
}

void setupIMU() {
  // IMU power and I2C setup
  pinMode(6, OUTPUT);
  digitalWrite(6, HIGH);
  Wire.begin(3, 4);
  delay(100);

  if (!lsm6dsox.begin_I2C()) {
    Serial.println("ERROR: IMU not found!");
    while(1) delay(10);
  }

  // Configure for motion tracking
  lsm6dsox.setAccelRange(LSM6DS_ACCEL_RANGE_4_G);    // ±4g range
  lsm6dsox.setGyroRange(LSM6DS_GYRO_RANGE_500_DPS);  // ±500 degrees/sec
  lsm6dsox.setAccelDataRate(LSM6DS_RATE_104_HZ);
```

```
  lsm6dsox.setGyroDataRate(LSM6DS_RATE_104_HZ);

  Serial.println("✓ IMU initialized");
}

void setupSDCard() {
  if (!SD.begin()) {
    Serial.println("WARNING: SD Card not found - logging disabled");
    return;
  }

  // Create new session file
  String filename = "/motion_" + String(millis()) + ".csv";
  dataFile = SD.open(filename, FILE_WRITE);

  if (dataFile) {
    dataFile.println("timestamp,accel_x,accel_y,accel_z,gyro_x,gyro_y,gyro_z,temp");
    dataFile.flush();
    Serial.println("✓ SD Card ready - " + filename);
  }
}

void setupWiFi() {
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");

  int attempts = 0;
  while (WiFi.status() != WL_CONNECTED && attempts < 20) {
    delay(500);
    Serial.print(".");
    attempts++;
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println();
    Serial.println("✓ WiFi connected!");
    Serial.print("IP: ");
    Serial.println(WiFi.localIP());
    udp.begin(udp_port);
  } else {
    Serial.println();
    Serial.println("WARNING: WiFi connection failed - streaming disabled");
  }
}

void loop() {
  unsigned long currentTime = millis();

  if (currentTime - lastSample >= SAMPLE_INTERVAL) {
    MotionData data = readMotionData();

    // Log to SD card
    logToSD(data);

    // Stream via WiFi
    streamData(data);

    lastSample = currentTime;
    packetCount++;

    // Status update every 5 seconds
    if (packetCount % 100 == 0) {
      Serial.println("Packets sent: " + String(packetCount));
    }
  }
}
```

```
MotionData readMotionData() {
  sensors_event_t accel, gyro, temp;
  lsm6dsox.getEvent(&accel, &gyro, &temp);

  MotionData data;
  data.accelX = accel.acceleration.x;
  data.accelY = accel.acceleration.y;
  data.accelZ = accel.acceleration.z;
  data.gyroX = gyro.gyro.x;
  data.gyroY = gyro.gyro.y;
  data.gyroZ = gyro.gyro.z;
  data.temp = temp.temperature;
  data.timestamp = millis() - sessionStart;

  return data;
}

void logToSD(MotionData data) {
  if (!dataFile) return;

  dataFile.print(data.timestamp); dataFile.print(",");
  dataFile.print(data.accelX, 3); dataFile.print(",");
  dataFile.print(data.accelY, 3); dataFile.print(",");
  dataFile.print(data.accelZ, 3); dataFile.print(",");
  dataFile.print(data.gyroX, 3); dataFile.print(",");
  dataFile.print(data.gyroY, 3); dataFile.print(",");
  dataFile.print(data.gyroZ, 3); dataFile.print(",");
  dataFile.println(data.temp, 1);

  // Flush every 10 samples to prevent data loss
  if (packetCount % 10 == 0) {
    dataFile.flush();
  }
}

void streamData(MotionData data) {
  if (WiFi.status() != WL_CONNECTED) return;

  // Create JSON packet for easy parsing
  String packet = "{";
  packet += "\"t\":" + String(data.timestamp) + ",";
  packet += "\"ax\":" + String(data.accelX, 3) + ",";
  packet += "\"ay\":" + String(data.accelY, 3) + ",";
  packet += "\"az\":" + String(data.accelZ, 3) + ",";
  packet += "\"gx\":" + String(data.gyroX, 3) + ",";
  packet += "\"gy\":" + String(data.gyroY, 3) + ",";
  packet += "\"gz\":" + String(data.gyroZ, 3) + ",";
  packet += "\"temp\":" + String(data.temp, 1);
  packet += "}";

  udp.beginPacket(host_ip, udp_port);
  udp.print(packet);
  udp.endPacket();
}
```

Configuration and Upload

1. Configure Network Settings

Update these lines in the code with your specific network information:

```
const char* ssid = "YOUR_WIFI_NAME";          // Your WiFi network name
const char* password = "YOUR_WIFI_PASSWORD";   // Your WiFi password
const char* host_ip = "192.168.1.100";        // Your computer's IP address
```

> **Finding Your Computer's IP Address:**
>
> **Windows:**
>
> Open Command Prompt, type `ipconfig`
>
> **Mac/Linux:**
>
> Open Terminal, type `ifconfig`
>
> Look for your local IP address (usually starts with 192.168.x.x)

2. Insert SD Card and Upload

1. Insert the micro SD card from your kit into the tinyCore
2. Connect via USB-C and select your board/port
3. Upload the code and open the Serial Monitor
4. Watch for startup messages:

```
✓ IMU initialized
✓ SD Card ready - /motion_1234567.csv
✓ WiFi connected!
IP: 192.168.1.105
```

Python Visualization Application

Create this Python script to visualize your motion data in real-time. Save it as `motion_visualizer.py`:

```
import socket
import json
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from collections import deque
import numpy as np

# Network settings (match your Arduino code)
UDP_IP = "0.0.0.0"  # Listen on all interfaces
UDP_PORT = 12345

# Data storage
max_points = 500
timestamps = deque(maxlen=max_points)
accel_data = {'x': deque(maxlen=max_points),
              'y': deque(maxlen=max_points),
              'z': deque(maxlen=max_points)}
gyro_data = {'x': deque(maxlen=max_points),
             'y': deque(maxlen=max_points),
             'z': deque(maxlen=max_points)}

# Setup UDP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((UDP_IP, UDP_PORT))
sock.settimeout(0.1)  # Non-blocking

# Setup plots
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))
fig.suptitle('tinyCore Motion Tracker - Live Data', fontsize=16)

def update_plot(frame):
    # Try to receive data
    try:
        data, addr = sock.recvfrom(1024)
        packet = json.loads(data.decode())

        # Store data
        timestamps.append(packet['t'] / 1000.0)  # Convert to seconds
        accel_data['x'].append(packet['ax'])
        accel_data['y'].append(packet['ay'])
        accel_data['z'].append(packet['az'])
        gyro_data['x'].append(packet['gx'])
        gyro_data['y'].append(packet['gy'])
        gyro_data['z'].append(packet['gz'])

    except (socket.timeout, json.JSONDecodeError):
        pass  # No data received, continue with existing data

    if len(timestamps) < 2:
        return

    # Clear and redraw plots
    ax1.clear()
    ax2.clear()

    time_array = list(timestamps)

    # Plot acceleration
    ax1.plot(time_array, list(accel_data['x']), 'r-', label='X-axis', linewidth=2)
    ax1.plot(time_array, list(accel_data['y']), 'g-', label='Y-axis', linewidth=2)
    ax1.plot(time_array, list(accel_data['z']), 'b-', label='Z-axis', linewidth=2)
    ax1.set_title('Acceleration (m/s²)')
    ax1.set_ylabel('Acceleration')
    ax1.legend()
    ax1.grid(True, alpha=0.3)
```

```
    # Plot gyroscope
    ax2.plot(time_array, list(gyro_data['x']), 'r-', label='X-rotation', linewidth=2)
    ax2.plot(time_array, list(gyro_data['y']), 'g-', label='Y-rotation', linewidth=2)
    ax2.plot(time_array, list(gyro_data['z']), 'b-', label='Z-rotation', linewidth=2)
    ax2.set_title('Gyroscope (rad/s)')
    ax2.set_xlabel('Time (seconds)')
    ax2.set_ylabel('Angular Velocity')
    ax2.legend()
    ax2.grid(True, alpha=0.3)

# Start animation
print(f"Listening for motion data on port {UDP_PORT}...")
print("Start your tinyCore motion tracker!")

ani = animation.FuncAnimation(fig, update_plot, interval=50, blit=False)
plt.tight_layout()
plt.show()
```

Running the Visualizer

1. **Install required Python packages:**

   ```
   pip install matplotlib numpy
   ```

2. **Run the visualization script:**

   ```
   python motion_visualizer.py
   ```

3. **Start moving your tinyCore** - you should see real-time graphs!

Testing and Experiments

Experiment Ideas

Try these movements and observe the data patterns:

1. Gravity Test

- Place tinyCore flat on table
- Z-axis acceleration should read ~9.8 m/s^2 (gravity!)
- Other axes should be near zero

2. Shake Test

- Shake the device back and forth
- Watch acceleration spikes in all directions
- Gyroscope should show rotational movement

3. Circle Drawing

- Draw smooth circles in the air
- Look for sine wave patterns in the data
- Try different orientations and speeds

4. Drop Test (Carefully!)

- Drop the tinyCore a few inches onto a soft surface
- You should see a brief period of zero acceleration (free fall!)

Understanding the Results

Acceleration Data:

- **Steady values around ±9.8:** Device orientation relative to gravity
- **Sharp spikes:** Quick movements or impacts
- **Smooth curves:** Rotational movements

Gyroscope Data:

- **Values near zero:** No rotation
- **Positive/negative peaks:** Rotation direction and speed
- **Smooth curves:** Consistent spinning motion

Data Analysis

Your SD card now contains CSV files with all your motion data! You can:

- Import into Excel/Google Sheets for analysis
- Use Python pandas for advanced processing
- Calculate total movement, detect patterns, or train ML models

What's Next?

You've built a complete IoT motion sensing system! Here are some ways to expand it:

Hardware Additions:

- Add external sensors via the QWIIC connectors
- Connect LEDs or buzzers for motion-triggered alerts
- Attach a battery for portable operation

Software Enhancements:

- Implement Kalman filtering for smoother data
- Add gesture recognition algorithms
- Create a web dashboard for remote monitoring
- Set up cloud data logging

Project Ideas:

- **Fitness Tracker:** Count reps, detect exercise types
- **Security System:** Motion-triggered alerts
- **Musical Interface:** Control sound with gestures
- **Physics Experiments:** Measure pendulum motion, free fall
- **IoT Sensor Network:** Multiple tinyCore devices reporting to central hub

Achievement Unlocked! 🎉

IoT Motion Tracker Complete!

You've successfully built and programmed a complete wireless motion sensing system. Time to show it off!

6. Troubleshooting

Common Issues and Solutions

Arduino IDE Setup Problems

Board Manager URL Issues:

- Verify the URL is entered correctly without extra spaces
- Check your internet connection
- Try restarting Arduino IDE

Board Not Found:

- Ensure you've installed both ESP32 and tinyCore board packages
- Check that your USB-C cable supports data (not just charging)
- Try a different USB port

Upload Failures:

- Verify Upload Mode is set to `UART0 / Hardware CDC`
- Ensure USB Mode is set to `Hardware CDC and JTAG`
- Try pressing and holding the BOOT button during upload

IMU Problems

IMU Not Found:

- Check that pin 6 is set HIGH (IMU power)
- Verify I2C initialization with `Wire.begin(3, 4)`
- Run an I2C scanner to detect the device address
- For legacy boards, ensure you're using the correct address (0x69)

Erratic Readings:

- Add delay after IMU initialization
- Check for loose connections if using breadboard
- Reduce sampling rate if data appears noisy

Motion Tracker Issues

WiFi Connection Problems:

- Double-check network credentials (case-sensitive)
- Ensure you're using a 2.4GHz network (ESP32 doesn't support 5GHz)
- Move closer to your router
- Check if your network requires special authentication

SD Card Not Working:

- Ensure the card is properly inserted
- Try reformatting as FAT32
- Check that the card isn't write-protected
- Verify the card is 32GB or smaller

Python Visualization Issues:

- Verify IP addresses match between Arduino and computer
- Ensure both devices are on the same network
- Check firewall settings aren't blocking UDP traffic
- Try running Python script as administrator

General Debugging Tips

1. **Use Serial Monitor:** Always check the Serial Monitor for error messages and status updates
2. **Check Connections:** Verify all physical connections are secure
3. **Power Issues:** Ensure adequate power supply via USB
4. **Library Versions:** Use the specified library versions for best compatibility
5. **Start Simple:** If complex code fails, try basic examples first

Getting Help

If you're still having trouble after trying these solutions:

- **Email Support:** support@mr.industries
- **Join Discord:** https://discord.gg/hvJZhwfQsF
- **Include Details:** When asking for help, include:
  - Error messages from Serial Monitor
  - Your operating system
  - Arduino IDE version
  - Exact steps that led to the problem

> **Congratulations!**
>
> You've completed the tinyCore setup guide and built your first IoT project. You're now ready to create amazing motion-sensing applications!
