Course schedule

- Package lectures: Hybrid lectures and tutorials every week.

  – Monday 9am - 11am   E3-06-08

  – Friday   4pm - 6pm   E1-06-04

- Lab and Lab test. 8 sessions (3 labs + project 5)

  – Thursday 9am - 12noon, E1-06-03

  – Date TBA

- Makeup session(s) and consultation sessions will be arranged if needed.

Assessments

- Lab and lab test   30%

- Project   30%

- Final exam   40%

Chapter 1
Introduction of IoT and Tools

Zhang Jianwen

elezhan@nus.edu.sg

Office: E4-05-21

Internet of things (IoT)

- Network of physical objects embedded with sensors, software, and technologies to connect and exchange data over the internet.

  - Extend the connectivity beyond standard devices to a diverse range of everyday objects.

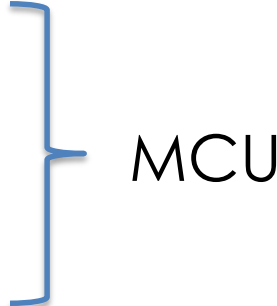 - Make them interact in a smart way .

Internet of things (IoT)

- Key Characteristics of IoT.

  – Connectivity

  – Data acquisition through sensors

  – Data processing

  – Automation and control for asset and actuators

  – Scalability

Internet of things (IoT)

- IoT components.

  – Sensors/Devices

  – Connectivity

  – Data Processing

  – User Interface

IoT framework

- Four main components.

  - End nodes: sensor node hardware

  - Network layer: communication protocols

  - Service layer: provide reliable link and data processing.

  - Application layer: integration service for data sharing.

IoT hardware

- Physical devices and components essential for the functioning of IoT systems.

  - Sensors and actuators

  - Processing data   MCU

  - Communication

IoT hardware

- IoT hardware

  - Sensors and actuators

  - Microcontrollers and microprocessors

  - Communication module

  - Power Supply

  - Edge Devices

  - Gateways

Example Applications of IoT

- Smart Homes: Connected appliances, lighting, security cameras, thermostats.

- Wearables: Fitness trackers, smartwatches, health monitoring.

- Industrial IoT (IIoT): Manufacturing optimization, equipment monitoring.

- Smart Cities: Traffic management, waste management, energy-efficient buildings.

IoT in Healthcare and Agriculture

- Healthcare: Remote patient monitoring, connected medical devices, telemedicine.

- Agriculture: Soil moisture sensors, automated irrigation, livestock monitoring.

Trends in IoT

- Edge computing

- AI and machine learning

- 5G connectivity

- Enhanced security

- Interoperability and standards

- Increased adoption in industrial IoT

Challenges in IoT

- Security and privacy

- Scalability

- Interoperability

- Data management

- Power consumption

- Standardization

Tools used in this course

- Git and GitHub

  - Refer and share codes

  - Submit and showcase you work in this course

- ESP32-S3 devkit and sensors

- Arduino IDE and supporting packages

Git and GitHub

- Version control, track your coding project progress and showcase.

- Git: local version control tool.

- GitHub: Web-based hosting platform interact with local Git.

- Why and how version control?

- Why web-based GitHub?

Git and GitHub

- Git workflow

  - Working directory

    - Untracked files

  - Staging area

    - Buffer area for files to be committed.

  - Git depository

    - Commit the changes permanently.

  - Master branch and other branches in a repo

Working Directory Staging Area Git Repository

add commit

Git and GitHub

- Install and configure Git

  - Install Git https://git-scm.com/downloads

  - Configure your Git (in command prompt)

    - User name

    - Email

    - Default editor Create a local Git depo

  - Convert the directory storing code as the project repo

Git and GitHub

- Git operations.

git init	git commit –m "[msg]"
git status	git log --oneline
git ls-files	git branch [branch name]
git add [filename]	git switch [branch name]
git rm --cache [filename]	git merge [branch to merge]
git reset –hard [commit hash]	git checkout [commit hash]

Git and GitHub

- Benefits of GitHub (or other Hubs) provided.

  – Create a repository (repo). Use the repo to store your code, share and collaborate with others.

  – Push your code from you local Git to the repo space on GitHub to share.

  – Showcase your study and work experience and achievement.

  – Evaluate others' work.

Git and GitHub

- Scenarios of GitHub usages.

  – Sign up an account.

  – Create a repository (repo).

  – Push your code from you local Git to GitHub or pull from GitHub to local Git.

  – Merge different branch

  – Collaborate with others

  – Showcase your work with readme.

Git and GitHub

– Push your code from you local Git to GitHub or pull from GitHub to local Git.

Push	Pull
git remote –v	git pull origin master
git remote add origin [url of the GitHub repo]	
git push –u origin master	

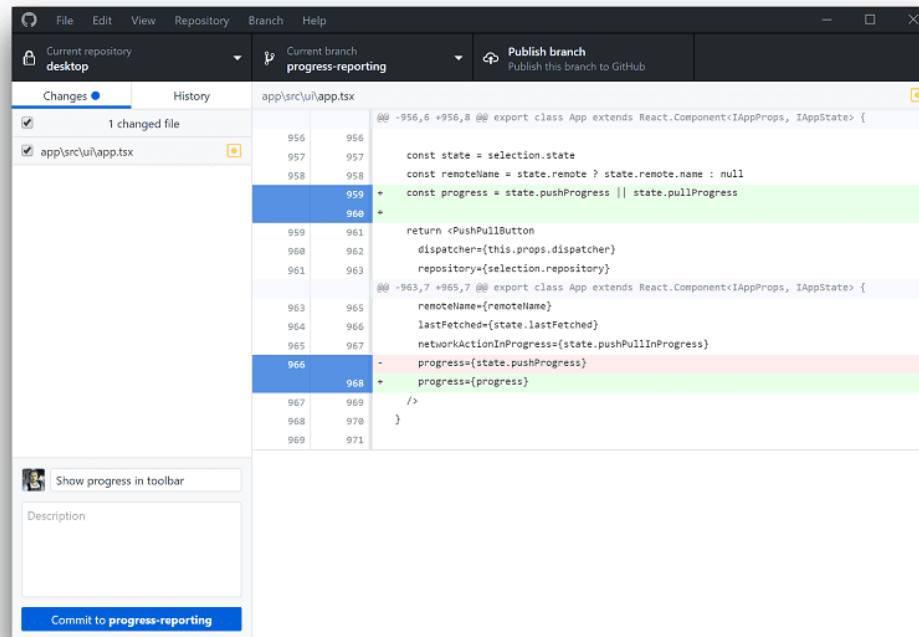# Git and GitHub

- Collaborate with others

  - Collaborator access by invitation as if in one's own GitHub repo

    Clone to local repo   git clone [url]

  - Pull request for non-collaborator.

    - Fork to replicate in your own Repo.

    - Clone to local Repo and contribute.

    - Pull to your GitHub Repo.

    - Raise a pull request to merge to the original .

    - Merge the changes to the original Repo.

Use Git and GitHub

- GitHub desktop: Graphical local Git tool for GitHub.

  - https://desktop.github.com/download/
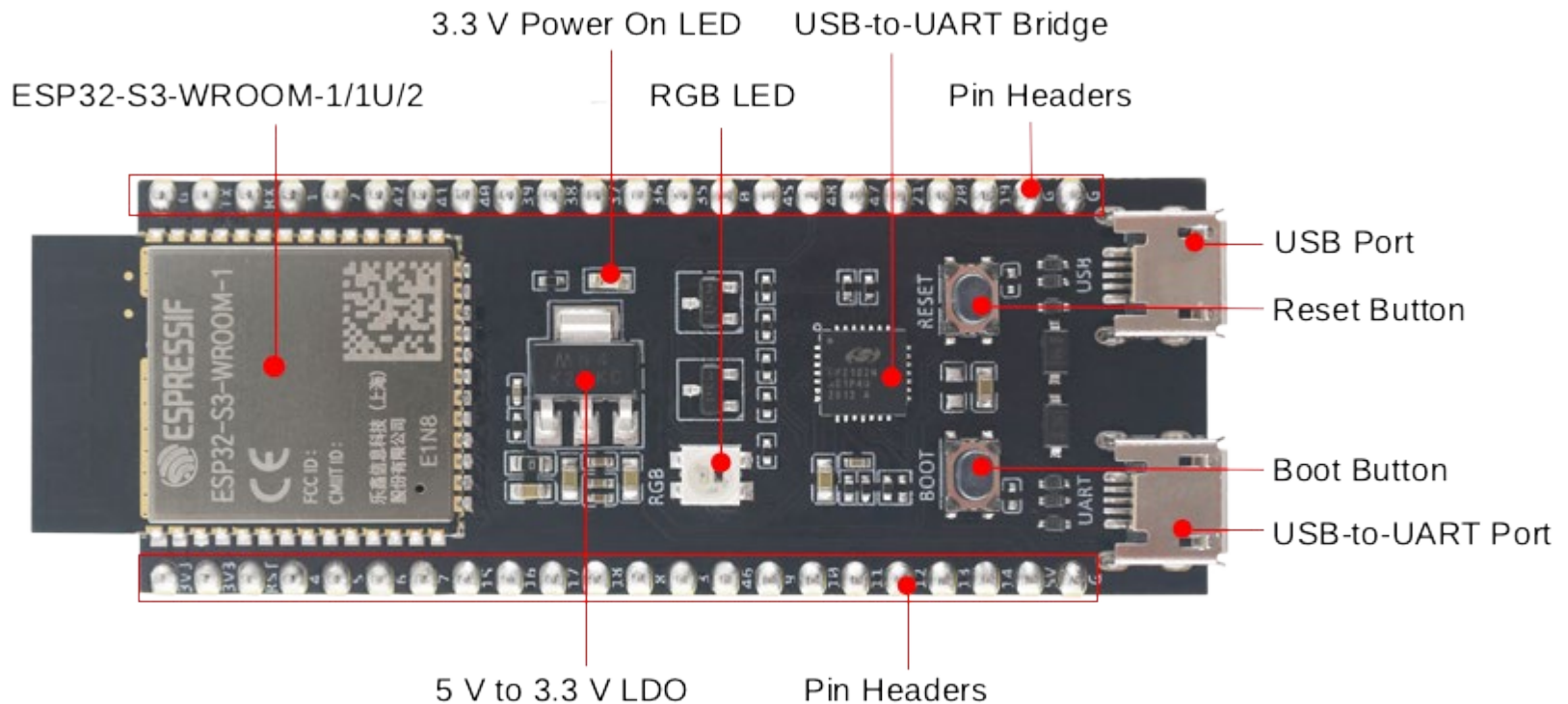
Use Git and GitHub

- Git operations.

- Functions of GitHub provided.

  – Create a repository (repo).

  Use the repo to store your code, share and collaborate with others

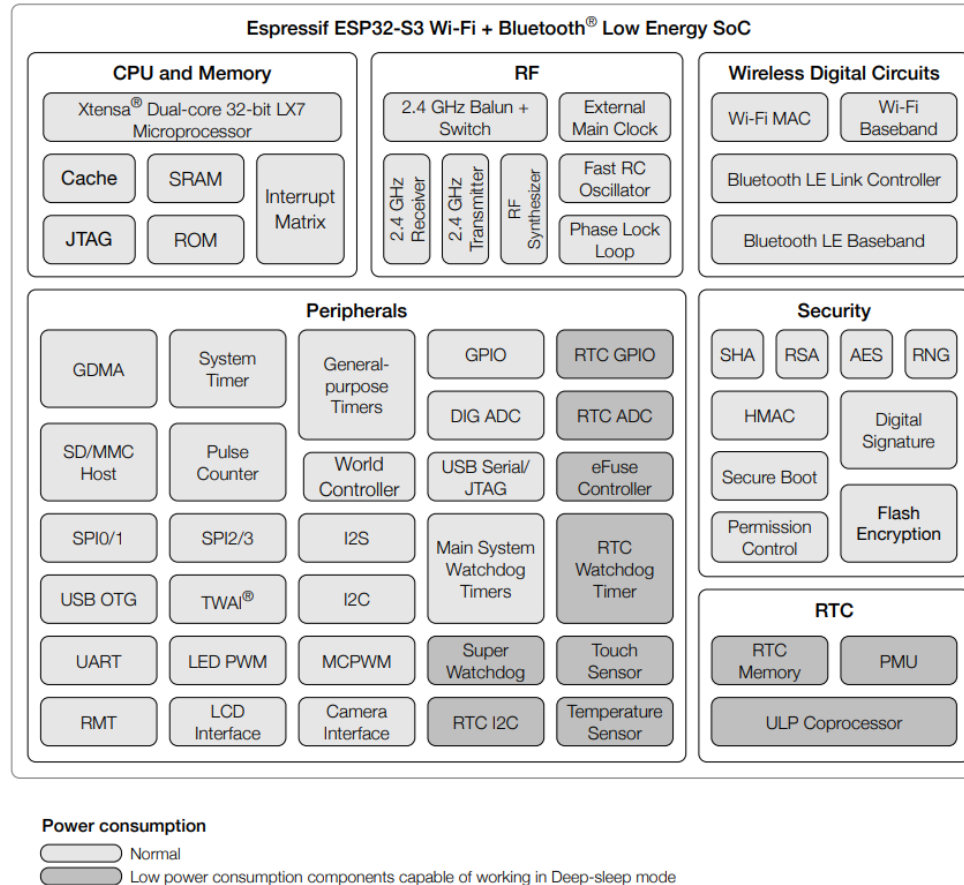 – Push your code from you local Git to the repo space on GitHub to share.

ESP32-S3 develop board

- ESP32-S3-DevKitC-1

ESP32-S3 develop board

- ESP32-S3-WROOM-1/1U/2 MCU module

Espressif ESP32-S3 Wi-Fi + Bluetooth® Low Energy SoC

ESP32-S3 develop board

- Processor cores: Dual core Xtensa LX7 up to 240MHz.

- Power supply: 3.3V and 5V with power regulator on board.

- USB-to-UART port and USB-to-UART bridge
  - UART (Universal Asynchronous Receiver-Transmitter)
    - Minimum 3 wires, TX, RX and GND, and 6 optional wires.
    - Relatively short distance and low rate
    - USB connector connecting to UART interface – CP2102
    - Communicate between MCU to computer and flash applications to MCU.
   - 5V power supply

ESP32-S3 develop board

- Boot button

- Reset button

- USB port: USB OTG, power supply, flash applications and debugging.

- RGB LED: Addressable RGB LED, driven by a GPIO pin.

- GPIO

ESP32-S3 develop board

- ESP32 is a powerful microcontroller with integrated Wi-Fi and Bluetooth.

- Suitable for IoT applications.

- Rich set of hardware resources.

ESP32-S3-DevKitC-1

ESPRESSIF
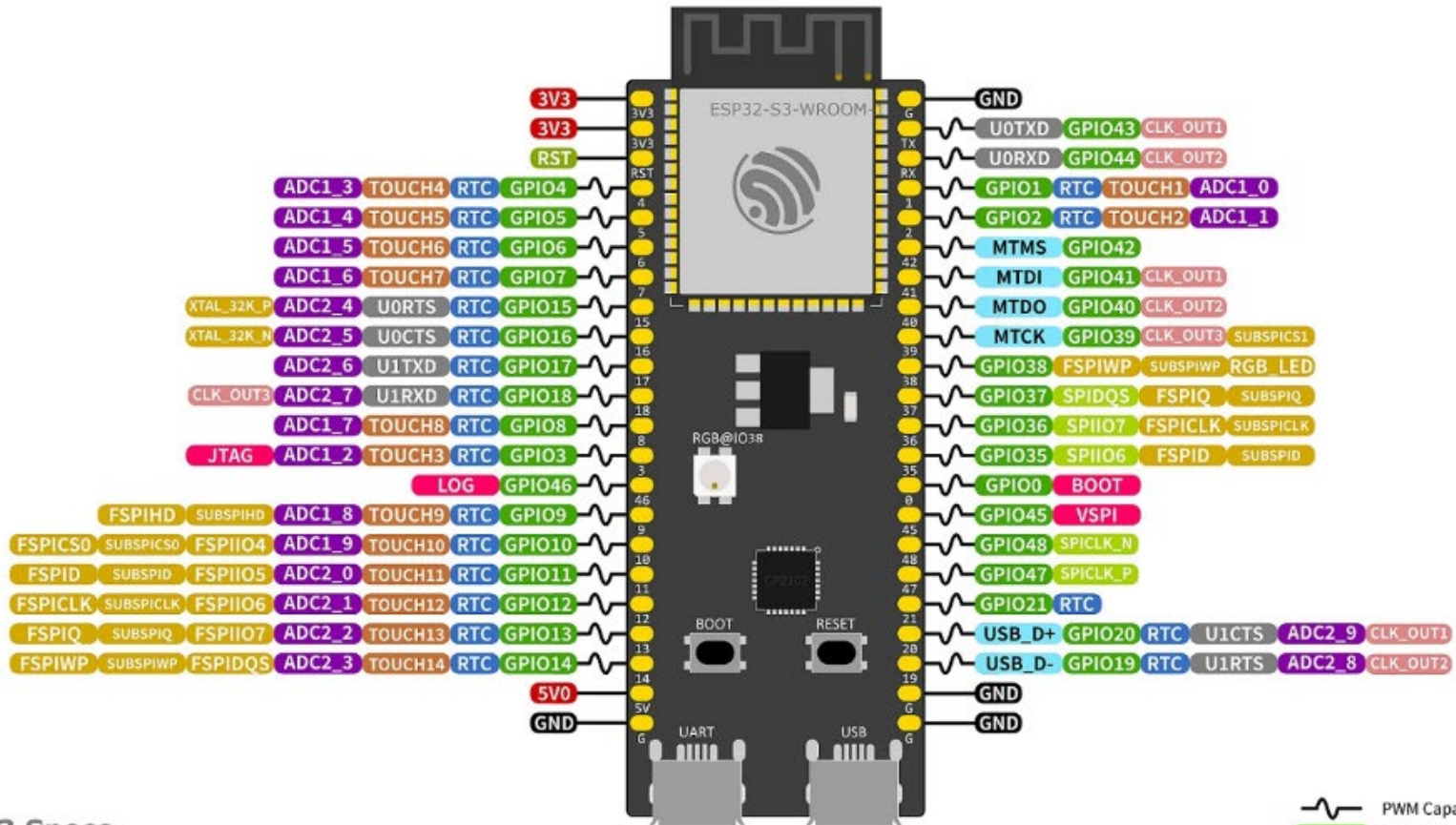
ESP32-S3 Specs
32-bit Xtensa® dual-core @240MHz
Wi-Fi IEEE 802.11 b/g/n 2.4GHz + BLE 5 Mesh
512 KB SRAM (16 KB SRAM in RTC)
384 KB ROM
45 GPIOs, 4x SPI, 3x UART, 2x I2C,
14x Touch, 2x I2S, RMT, LED PWM, USB-OTG,
TWAI®, 2x 12-bit ADC, 1x LCD interface, DVP

Legend:
- PWM Capable Pin
- GPIOX — GPIO Input and Output
- JTAG/USB — JTAG for Debugging and USB
- ADCX_CH — Analog-to-Digital Converter
- TOUCHX — Touch Sensor Input Channel
- OTHER — Other Related Functions
- SERIAL — Serial for Debug/Programming
- STRAP — Strapping Pin Functions
- RTC — RTC Power Domain (VDD3P3_RTC)
- GND — Ground
- PWD — Power Rails (3V3 and 5V)
- MISC — Miscellaneous/SPI functions
- CLK_OUTx — Clock Output

ESP32-S3 develop board

- ESP32 is a powerful microcontroller with integrated Wi-Fi and Bluetooth.

- Suitable for IoT applications.

- Rich set of hardware resources.

Arduino IDE

- Integrated development environment (IDE) for Arduino boards and other compatible boards

  - Support code editing, compilation, flash and debugging.

  - Community and industry develop various libraries

    - There are drivers for various peripherals

    - Libraries for various developing boards with different MCU, including ESP32-S3-DevKitC-1.

  - Programming language is a simplified C++ → Sketches

    - Support most of the useful features of C++, including pointers.

Why choose Arduino IDE?

- Arduino IDE is a familiar interface to many of us.

- Arduino board family are already used in IoT

- Arduino IDE supports many other developing board.

- Programming language: sketch = mixture of C and C++.

Configuring GPIOs in Arduino IDE

- Use pinMode(), digitalWrite(), and digitalRead() functions.

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

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

GPIO Interrupts

- Handle asynchronous events.

- Use attachInterrupt() function.

```
void IRAM_ATTR handleInterrupt() {
  // Interrupt service routine
}

void setup() {
  pinMode(2, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(2), handleInterrupt, FALLING);
}

void loop() {
  // Main loop code
}
```

Example: Button Control

- Control an LED with a button press.

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(2, INPUT_PULLUP);
}

void loop() {
  if (digitalRead(2) == LOW) {
    digitalWrite(LED_BUILTIN, HIGH);
  } else {
    digitalWrite(LED_BUILTIN, LOW);
  }
}
```
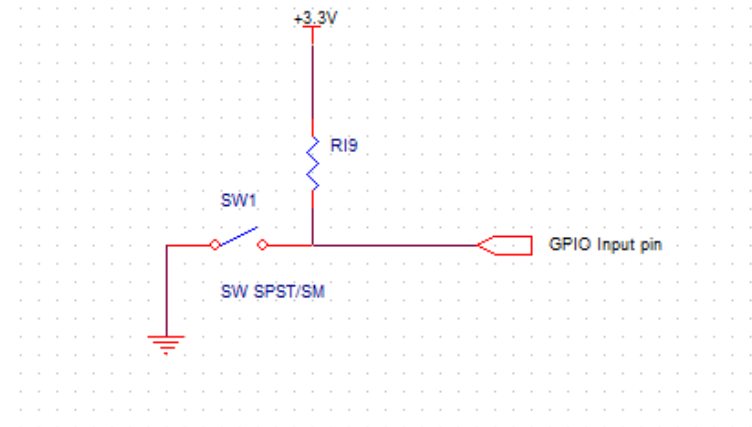
Configuring I2C in Arduino IDE

- Use Wire.begin(), Wire.requestFrom(), and

  Wire.write() functions.

```
void setup() {
  Wire.begin();
}


void loop() {
  Wire.beginTransmission(0x68);//device with i2C addr 0x68
  Wire.write(0x00);
  Wire.endTransmission();
  Wire.requestFrom(0x68, 1);
  while (Wire.available()) {
    char c = Wire.read();
  }
}
```

Default:
SDA (Data Line): GPIO 8
SCL (Clock Line): GPIO 9

```
void setup() {
  Serial.begin(115200);
  Wire.begin();
}

void loop() {
  Wire.beginTransmission(0x68);
  Wire.write(0x00);
  Wire.endTransmission();
  Wire.requestFrom(0x68, 2);
  while (Wire.available()) {
    int data = Wire.read() << 8 | Wire.read();
    Serial.println(data);
  }
  delay(1000);
}
```

Configuring SPI in Arduino IDE

- 2 SPI are available: HSPI and VSPI for developers.

```
void setup() {
  SPI.begin();
}

void loop() {
  digitalWrite(SS, LOW);
  SPI.transfer(0x00);
  digitalWrite(SS, HIGH);
  delay(1000);
}
```

MOSI (Master Out Slave In): GPIO 13
MISO (Master In Slave Out): GPIO 12S
CLK (Clock): GPIO 14
SS/CS (Slave Select/Chip Select): GPIO 15

Configuring ADC in Arduino IDE

- Analog-to-Digital Converter.

- Used to read analog signals.

- Multiple ADC channels.

ADC1 Channels:
ADC1_CH0: GPIO 36
ADC1_CH1: GPIO 37
ADC1_CH2: GPIO 38
ADC1_CH3: GPIO 39
ADC1_CH4: GPIO 32
ADC1_CH5: GPIO 33
ADC1_CH6: GPIO 34
ADC1_CH7: GPIO 35

```
void setup() {
  Serial.begin(115200);
}

void loop() {
  int value = analogRead(34);
  float voltage = value * (3.9 / 4095.0);
  float temperature = (voltage - 0.5) * 100;
  Serial.println(temperature);
  delay(1000);
}
```

EE4216

Configuring PWM in Arduino IDE

- Use ledcSetup(), ledcAttachPin(), and ledcWrite() functions.

```
void setup() {
  ledcSetup(0, 5000, 8);       // Configure LEDC channel 0 with a 5 kHz frequency and 8-bit resolution
  ledcAttachPin(2, 0);          // Attach GPIO 2 to LEDC channel 0
}

void loop() {
  for (int duty = 0; duty <= 255; duty++) {
    ledcWrite(0, duty);       // Set the duty cycle for channel 0 (0 to 255)
    delay(10);                 // Wait for 10 milliseconds
  }
  for (int duty = 255; duty >= 0; duty--) {
    ledcWrite(0, duty);       // Set the duty cycle for channel 0 (255 to 0)
    delay(10);                 // Wait for 10 milliseconds
  }
}
```

Configuring Wi-Fi in Arduino IDE

- Use WiFi.begin(), WiFi.status(), and WiFiClient functions.

```cpp
#include <WiFi.h>

void setup() {
  Serial.begin(115200);
  WiFi.begin("SSID", "PASSWORD");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting...");
  }
  Serial.println("Connected!");
}

void loop() {
  // Main loop code
}
```