CEC Continuing Education Center

DesignNews

Designing Embedded Systems using the ESP32

DAY 2 : Setting up and Exploring the SDK

Sponsored by

Digi-Key ELECTRONICS

informa markets

Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Group Chat’ by maximizing the chat widget in your dock.
- Submit questions for the lecturer using the Q&A widget. They will follow-up after the lecture portion concludes.

Course Sessions

- Introduction to the ESP32 Wi-Fi Module
- **Setting up and Exploring the SDK**
- Programming and Writing the First Application
- It's all about Wi-Fi
- Jump-Starting Cloud Connectivity Applications with Amazon FreeRTOS

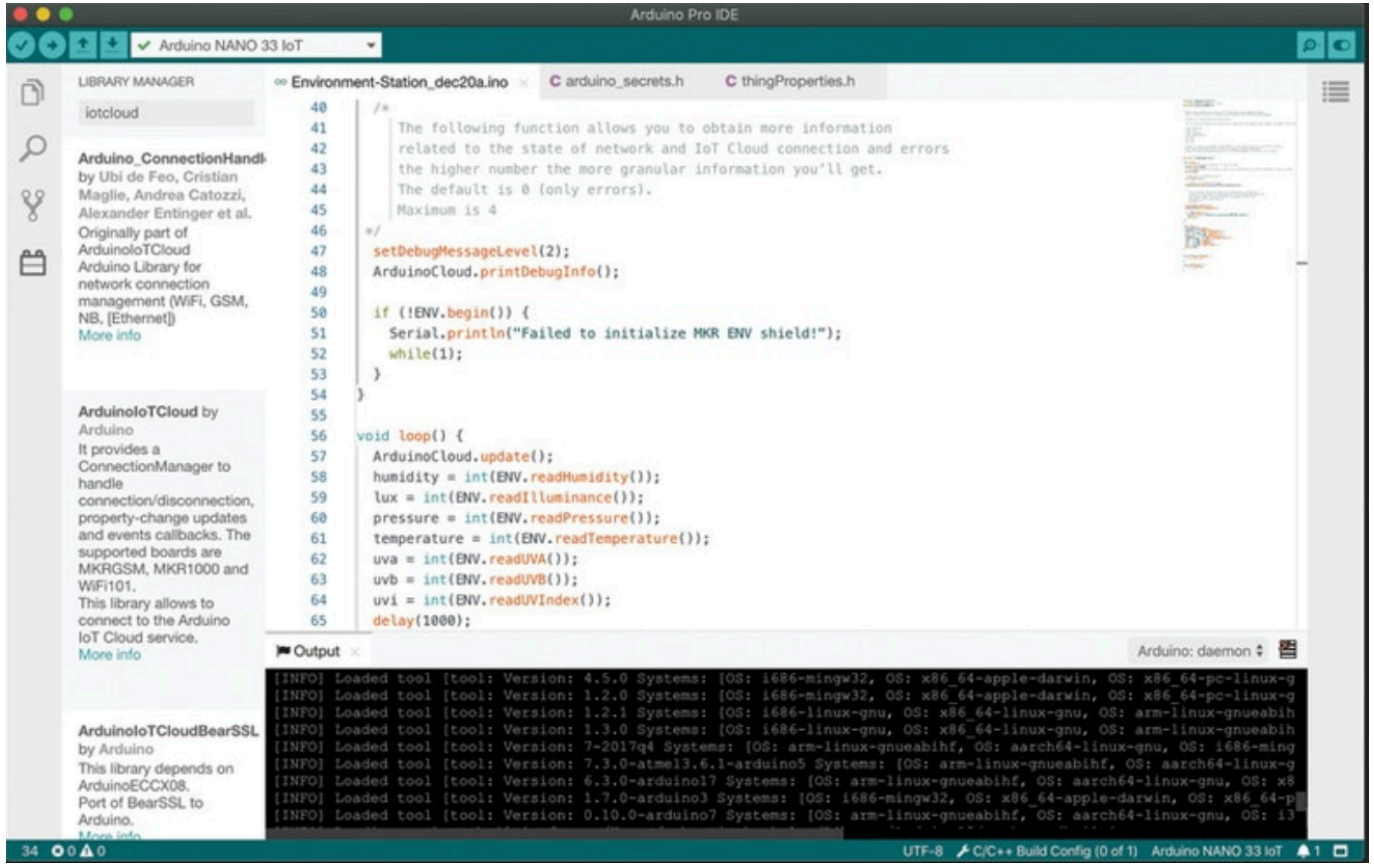
Arduino IDE

Pros

- Fast to setup
- Free
- Plenty of examples

Cons

- Slow compile times
- Rudimentary toolchain



MicroPython

Pros

- Fast to setup
- Free
- Quickly write apps in Python
- No low-level languages

Cons

- MicroPython is ~400 kB
- Features are not all supported

Libraries and external classes →

Definitions and Initialization →

Primary program loop →

```
# main.py
import pyb

# define LED color constants
LED_RED = 1
LED_GREEN = 2
LED_YELLOW = 3
LED_BLUE = 4

# Defines the primary loop delay
DELAY_1000MS = 1000

# Create an Led object assigned to the green LED
Led = pyb.LED(LED_GREEN)

# Main execution loop
# Toggle the LED every
while True:
    Led.toggle()
    pyb.delay(DELAY_1000MS)
```

Amazon FreeRTOS

Pros

- Easy cloud connectivity
- Flexible dev environments
- Widely supported

Cons

- ESP-IDF synchronization
- Growing code size

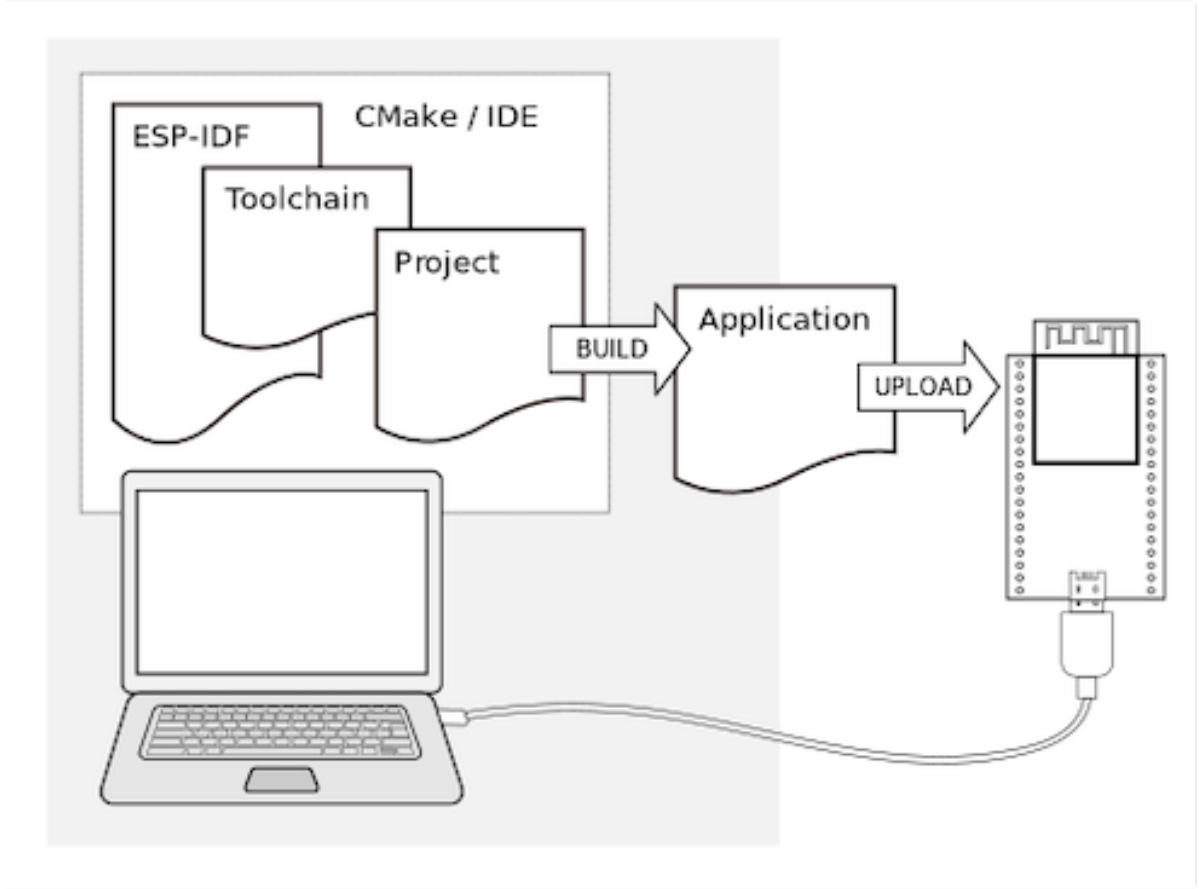
ESP-IDF

Pros

- Comprehensive device support
- Professional developer focused
- Integrated framework

Cons

- Can be “complex” to setup



Which development environment are you most interested in trying?

- Arduino
- MicroPython
- Amazon FreeRTOS
- ESP-IDF

ESP-IDF Setup

1) Select your OS

Windows

Linux

Mac OS

ESP-IDF Setup

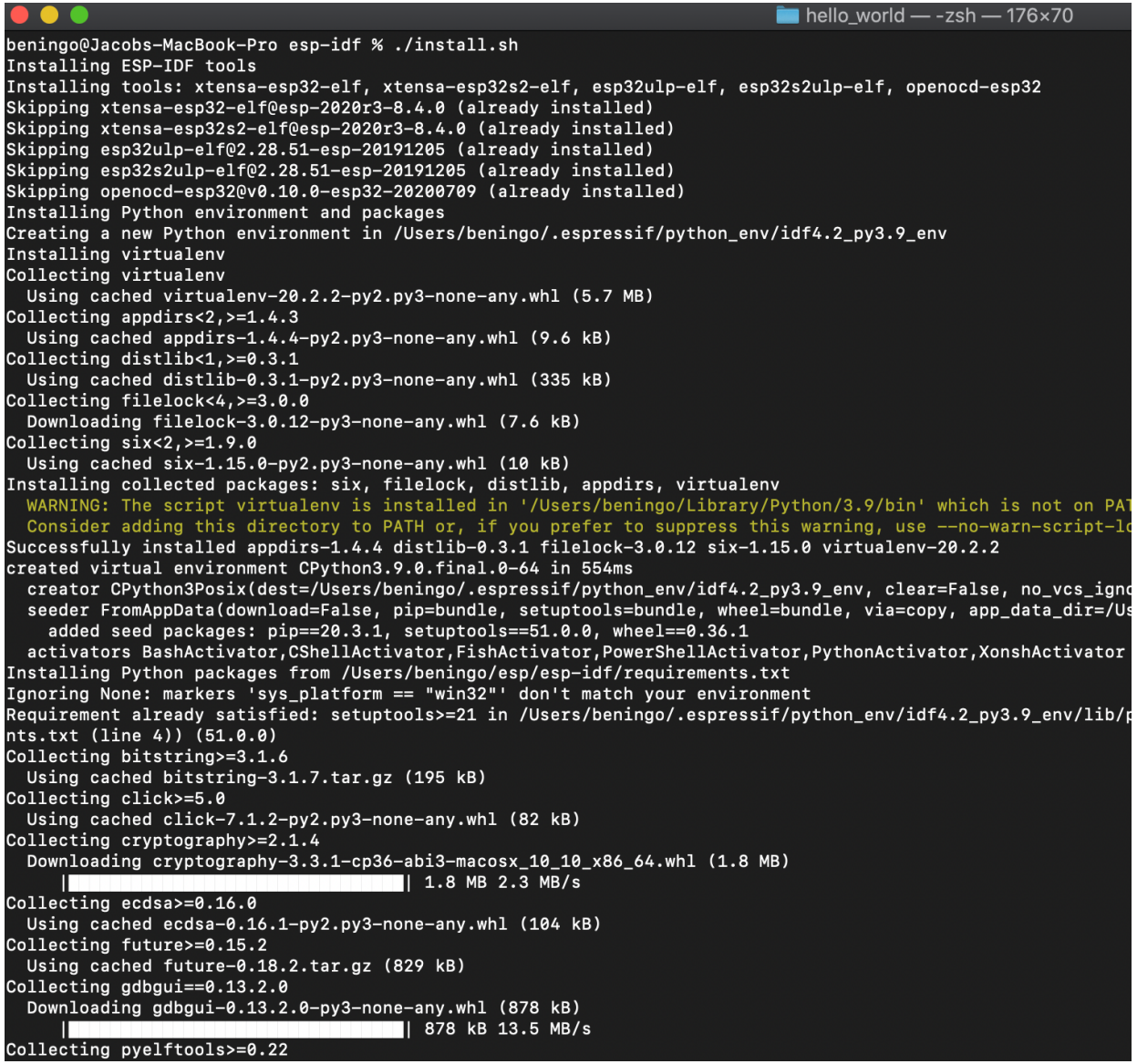
2) Get ESP-IDF

```
mkdir -p ~/esp
cd ~/esp
git clone –recursive https://github.com/espressif/esp-idf.git
```

ESP-IDF Setup

3) Set up the tools

cd ~/esp/esp-idf
./install.sh



ESP-IDF Setup

Package Issues:

You need to use Python 3. On MacOS, execute:

```
brew install python
ls -l /usr/local/bin/python*
ln -s -f /usr/local/bin/python3.9 /usr/local/bin/python
python --version
```

ESP-IDF Setup

4) Set up environment variables

. $HOME/esp/esp-idf/export.sh

You can also add to your terminal profile:

alias get_idf='. $HOME/esp/esp-idf/export.sh'

Did you follow along and install the toolchain?

- Yes
- No
- No but I plan to for tomorrow

ESP-IDF Overview

- Bluetooth
- Networking
- Peripheral
- Protocols
- Provisioning
- Storage
- System
- Configuration

ESP-IDF UART API

```
const int uart_num = UART2; uart_config_t uart_config =
{ .baud_rate = 115200,
  .data_bits = UART_DATA_8_BITS,
  .parity = UART_PARITY_DISABLE,
  .stop_bits = UART_STOP_BITS_1,
  .flow_ctrl = UART_HW_FLOWCTRL_CTS_RTS,
  .rx_flow_ctrl_thresh = 122, };

// Configure UART parameters ESP_ERROR_CHECK(uart_param_config(uart_num, &uart_config));

// Set UART pins(TX: IO16 (UART2 default), RX: IO17 (UART2 default), RTS: IO18, CTS: IO19)
ESP_ERROR_CHECK(uart_set_pin(UART_NUM_2, UART_PIN_NO_CHANGE, UART_PIN_NO_CHANGE, 18, 19));
```

ESP-IDF UART Example

Parameter to Configure	Function
Baud rate	`uart_set_baudrate()`
Number of transmitted bits	`uart_set_word_length()` selected out of `uart_word_length_t`
Parity control	`uart_set_parity()` selected out of `uart_parity_t`
Number of stop bits	`uart_set_stop_bits()` selected out of `uart_stop_bits_t`
Hardware flow control mode	`uart_set_hw_flow_ctrl()` selected out of `uart_hw_flowcontrol_t`
Communication mode	`uart_set_mode()` selected out of `uart_mode_t`

ESP-IDF UART Example

```
// Setup UART buffered IO with event queue
const int uart_buffer_size = (1024 * 2);
QueueHandle_t uart_queue; // Install UART driver using an event queue here
ESP_ERROR_CHECK(uart_driver_install(UART2, uart_buffer_size, \ uart_buffer_size, 10, &uart_queue, 0));

// Write data to UART.
char* test_str = "This is a test string.\n";
uart_write_bytes(uart_num, (const char*)test_str, strlen(test_str));
```

Do you plan to follow along live for the first application development tomorrow?

- Yes
- No

Thank you for attending

Please consider the resources below:

- www.beningo.com
  - Blog, White Papers, Courses
  - Embedded Bytes Newsletter
    - http://bit.ly/1BAHYXm

From www.beningo.com under

- Blog > CEC – Designing Embedded Systems using the ESP32

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

Digi-Key ELECTRONICS

informa markets