Topic 15

Using MicroPython on ESP32

Professor Peter YK Cheung
Dyson School of Design Engineering

URL: www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/
E-mail: p.cheung@imperial.ac.uk
ESP32 IoT Microcontroller (1)

- **Radio**
- **Bluetooth**
- **Wi-Fi**
- **CPU core & memory**
- **Embedded flash memory**
- **Peripheral Interface**
- **Real-time clock & Low Power processor unit**
- **Cryptographic Hardware**

- **ESPD32**
- **IoT**
- **Microcontroller (1)**
- **Radio**
- **CPU core & memory**
- **Peripheral Interface**
- **Embedded flash memory**

- **Radio**
- **Bluetooth**
- **Wi-Fi**
- **CPU core & memory**
- **Embedded flash memory**
- **Peripheral Interface**
- **Real-time clock & Low Power processor unit**
- **Cryptographic Hardware**

- **ESP32**
- **IoT**
- **Microcontroller (1)**
- **Radio**
- **CPU core & memory**
- **Peripheral Interface**
- **Embedded flash memory**
- **Real-time clock & Low Power processor unit**
- **Cryptographic Hardware**
ESP32 CPU Core & Memory

Core and memory

Xtensa LX6 microprocessor
32-bit; dual-core or single-core

ROM
Read-only memory

SRAM
Static random-access mem.
ESP32 RTC and Power Management Sub-system

**RTC and low-power management subsystem**

- **PMU** Power management unit
- **Ultra-low-power co-processor**
- **Recovery memory**
ESP32 Crypto Hardware

Cryptographic hardware acceleration

- **RSA**: Rivest-Shamir-Adleman
- **SHA**: FIPS PUB 180-4
- **RNG**: Random number gen.
- **AES**: FIPS PUB 197
ESP32 wireless links
ESP32 Peripheral Interfaces & SPI RAM

- Embedded flash memory
  - Included in ESP32-PICO-D4 system-in-package QFN module

**Peripheral interfaces**
- \( \text{I}^2\text{C} \)
  - Inter-Integrated Circuit
- SDIO
  - Secure Digital Input Output
- CAN
  - Controller Area Network
- IR
  - Infrared
- Temperature sensor
  - Internal; range of \(-40^\circ\text{C}\) to \(125^\circ\text{C}\)
- DAC
  - Digital-to-analog converter

**SPI**
- Serial Peripheral Interface

- \( \text{I}^2\text{S} \)
  - Inter-IC Sound
- UART
  - Universal async. receiver-transmitter
- ETH
  - Ethernet MAC
- PWM
  - Pulse-width modulation
- Touch sensors
  - Ten capacitive-sensing inputs
- SAR ADC
  - Successive approx. analog-to-digital conv.
Heltec ESP32 module

Note:
*GPIO input only
*ADC preamplifier
GPIOs are 3.3V tolerant only
ESP32 with MicroPython (uPy)

PyCharm IDE

Flash ESP32 via esptool

MicroPython firmware in flash ROM
- machine
- esp
- network
- sys
- esp32
- neopixel
- time
- os
- math

On-chip ROM for programs
- boot.py
- main.py
- oled.py
- hello.py
- rotary.py

Directory

Editor

REPL

uPy terminal
Lab 4A – Setting up the MicroPython environment

- **Task 1**: Install CP2102 drive driver
- **Task 2**: Install esptool
- **Task 3**: Erase and flash uPy to ESP32
- **Task 4**: Install PyCharm IDE with uPy plug-in
- **Tasks 5 & 6**: Explore uPy via REPL
- **Tasks 7**: Flashing "Hello world!" onto ESP32
- **Task 8**: Display on OLED
MicroPython documentation

Welcome! This is the documentation for MicroPython v1.12, last updated 05 Jun 2020.

MicroPython runs on a variety of systems and hardware platforms. Here you can read the general documentation which applies to all systems, as well as specific information about the various platforms - also known as ports - that MicroPython runs on.

General documentation for MicroPython:

- Library Reference
- MicroPython libraries and modules
- Language Reference
- information about MicroPython specific language features
- MicroPython Differences
- MicroPython operations which differ from CPython
- License
- MicroPython license information

References and tutorials for specific platforms:

- Quick reference for the ESP32
- pinout for ESP32-based boards, snippets of useful code, and a tutorial
MicroPython Library Functions

Python standard lib

- **math** – mathematical functions (e.g. sin, pi)
- **sys** – system specific functions (e.g. sys.argv)

MicroPython and ESP32 specific lib

- **machine** – functions related to processor itself
- **esp** – functions related to the board

- **time** – functions related to hardware timer

Various **class libraries** to drive peripherals
### pyb - Class Library

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class PWM</strong></td>
<td>PWM signal generation</td>
</tr>
<tr>
<td><strong>class ADC</strong></td>
<td>analog to digital conversion</td>
</tr>
<tr>
<td><strong>class DAC</strong></td>
<td>digital to analog conversion (2 channels)</td>
</tr>
<tr>
<td><strong>class LED</strong></td>
<td>LED objects to control on board LEDs</td>
</tr>
<tr>
<td><strong>class Pin</strong></td>
<td>control I/O pins</td>
</tr>
<tr>
<td><strong>class I2C</strong></td>
<td>control I2C interface</td>
</tr>
<tr>
<td><strong>class Timer</strong></td>
<td>control hardware timers</td>
</tr>
<tr>
<td><strong>class SPI</strong></td>
<td>control SPI interface</td>
</tr>
</tbody>
</table>