

Embedded Swift Workshop

ARCtic Conference 2025

Frank Lefebvre

Introduction

- Subset of the Swift language
 - Small runtime
 - No runtime reflection
 - Restrictions on existential types
 - Dynamic heap allocations can be disabled
- Work in progress

Supported Architectures

- RISC-V (ESP32)
- STM32
- ARM32 (nRF52840, Raspberry Pi Pico)
- ARM64 (Raspberry Pi 4b/5)
- PowerPC (Freescale)

The ESP32 Family

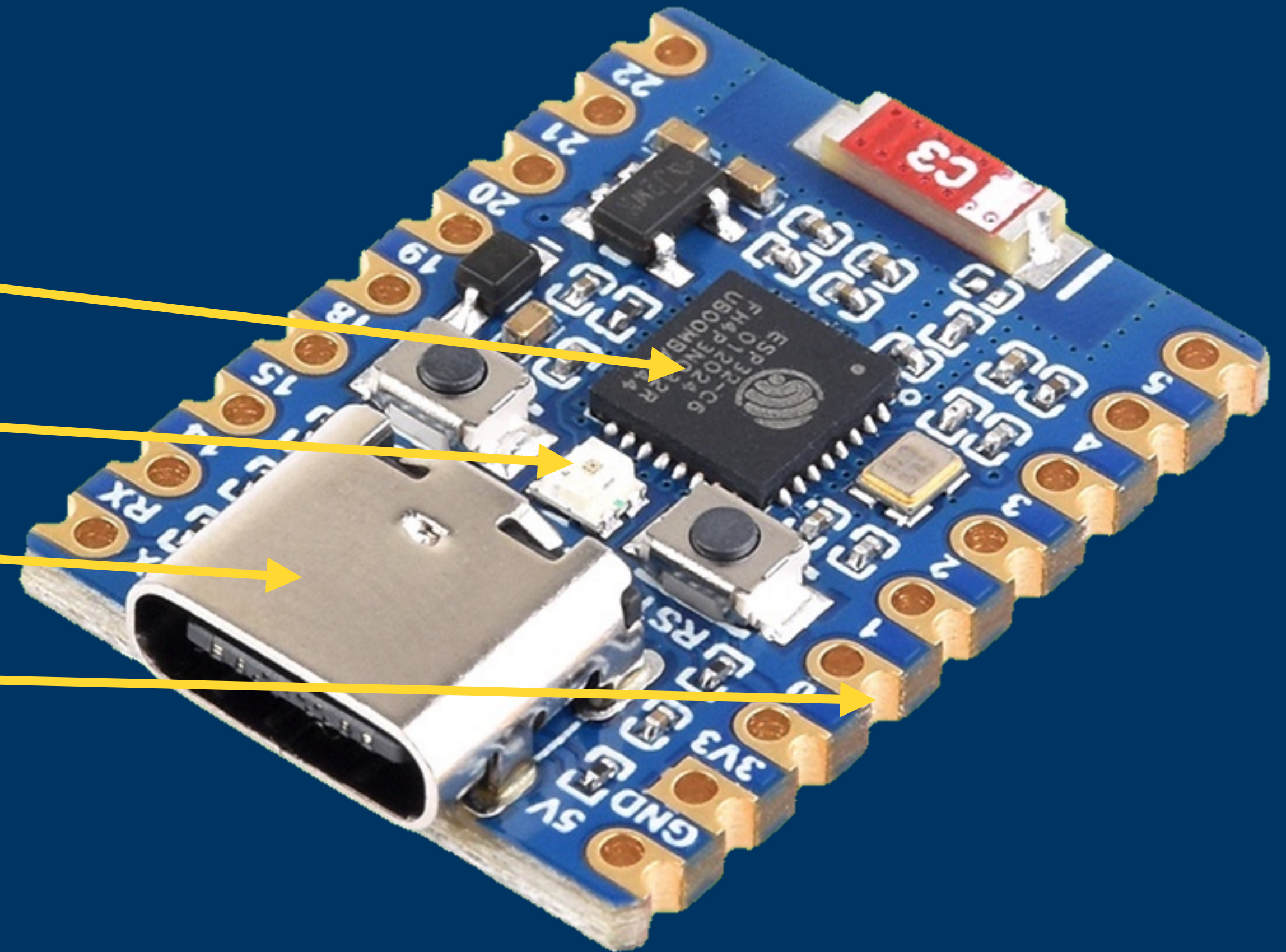
Series	Architecture	Wi-Fi	Bluetooth	Ethernet
ESP32-S	Xtensa (single core)	✓	✓ (ESP32-S3)	
ESP32-D	Xtensa (dual core)	✓		
ESP32-C	RISC-V (single core)	✓	✓	
ESP32-H	RISC-V (single core)		✓	
ESP32-P	RISC-V (dual core)			✓

ESP32-C6

- Single-Core 160 MHz RISC-V CPU (+ low-power core)
- 512KB RAM
- Up to 16MB Flash
- Wi-Fi 6 (2.4 GHz)
- Bluetooth LE 5.3
- Threads, Zigbee

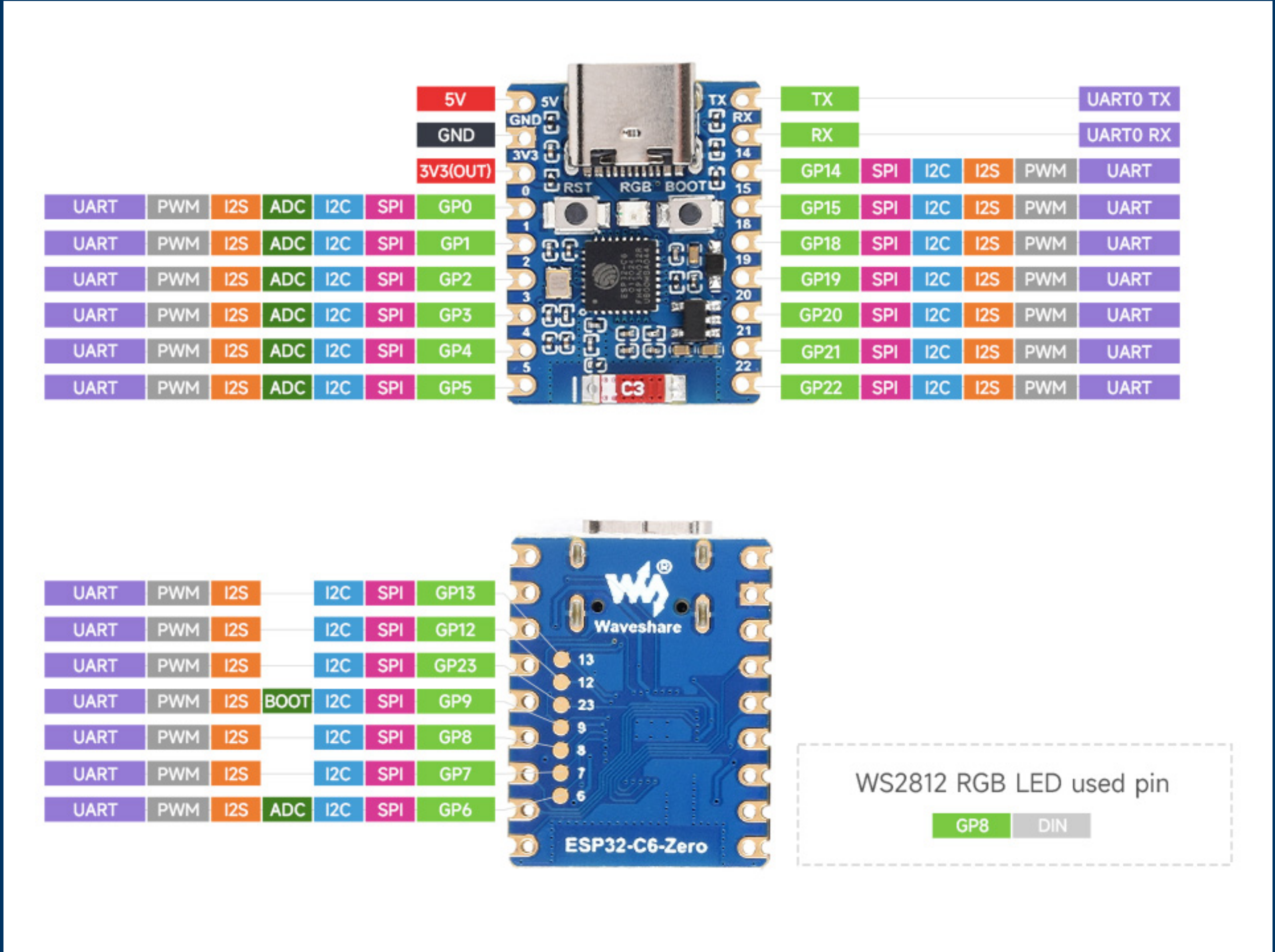
Waveshare ESP32-C6 development board

- ESP32-C6 (4MB Flash)
- RGB LED
- USB-C power+data
- 20 programmable IO pins



Development Board I/O

Flexible configuration



Tools & Resources

- Swift Toolchain: development snapshot
- Visual Studio Code
- ESP-IDF
- CMake, Ninja

Hands-on

Setup

- Download and install
 - Swift Toolchain
 - Visual Studio Code
 - ESP-IDF plugin for VS Code
 - Swift plugin for VS Code
- Build and run "hello world" project

Blinking LED

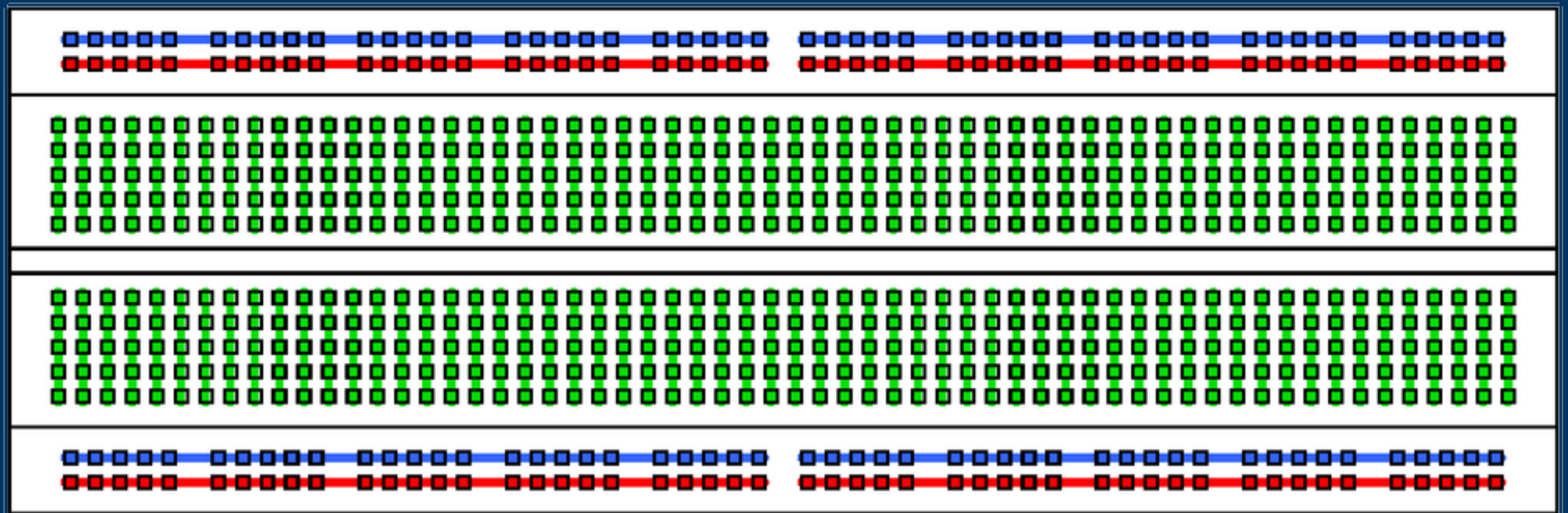
Basic GPIO

- GPIO pin configuration by software
- Output mode
 - 0V, 3.3V
 - 20 mA max (200 mA total)

LED

- Characteristic voltage: 1.8V (red/green), 2.5V (blue)
- 20 mA max
- Resistor: 330 Ω

Solderless Breadboard



Hands-on

Blinking LED

- Connect LED & resistor
- Open 00-Start-Here project with VS Code
- Change settings
- Build and run

Troubleshooting

Troubleshooting

Build & Install

- `$PATH`
 - `idf.py`
- `toolchain`
 - `swift -version`
- `connection`
 - `ls /dev/tty.*`
 - `echo $ESPPORT`

Troubleshooting

Runtime

- `print()` debugging
 - string interpolation limitations
- LED debugging
- Core dump analysis
 - swift demangle
 - use the embedded toolchain

Swift-C Interoperability 101

Using C Types from Swift

Swift-C Interoperability

- BridgingHeader.h
- Simple types
- Simple functions
- #define macros

Memory Layout

- `MemoryLayout<Type>.size`
- `MemoryLayout<Type>.stride`
- `MemoryLayout<Type>.alignment`

Typed Pointers

- `UnsafePointer<Type>`
- `UnsafeMutablePointer<Type>`
- access to payload: `pointee`
- Heap allocation: `allocate(capacity:)`, `deallocate()`
- Temporary use as pointer: `withUnsafe[Mutable]Pointer { ptr in ... }`
 - Valid only inside closure

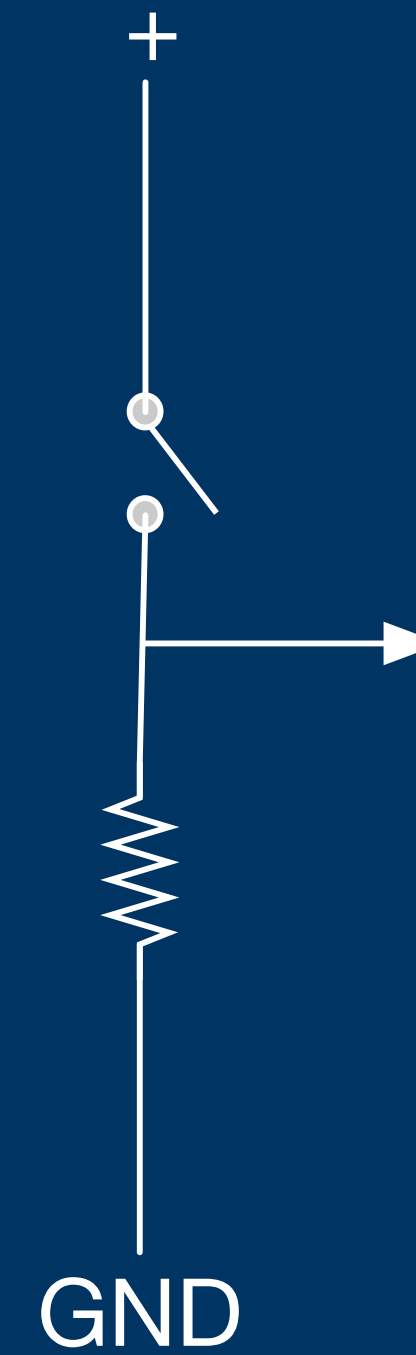
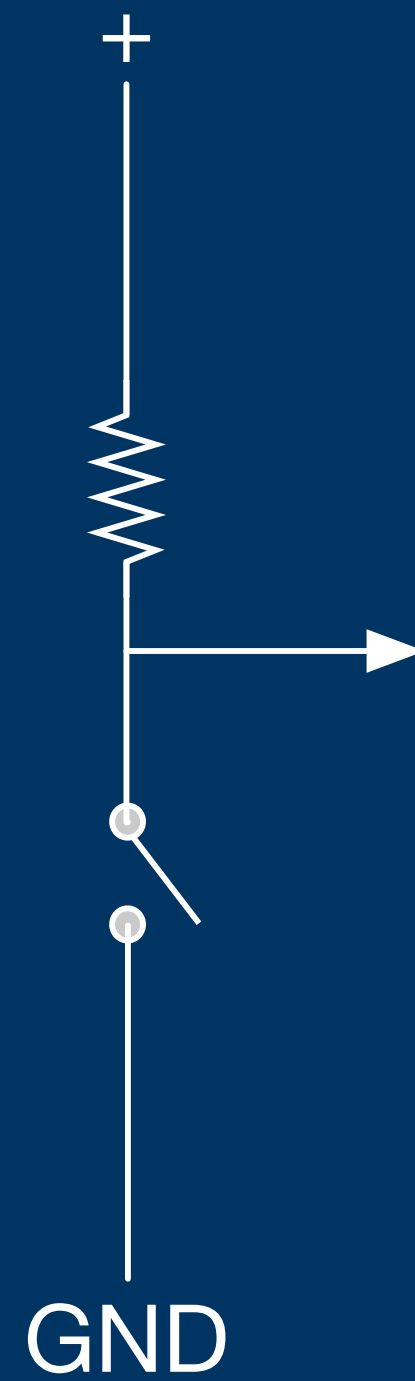
Opaque Pointers

- C: `void *`
- Swift: `UnsafeMutableRawPointer?`
- Access to typed data
 - `assumingMemoryBound(to: Type.self) -> UnsafeMutablePointer<Type>`

Simple Input

GPIO Input Setup

- Switch: opened, closed
- Pull-up, pull-down



Hands-on

Simple input

- Connect the switch
- Create Input class
- Update main
- Test

Interrupts

Interrupts on ESP32

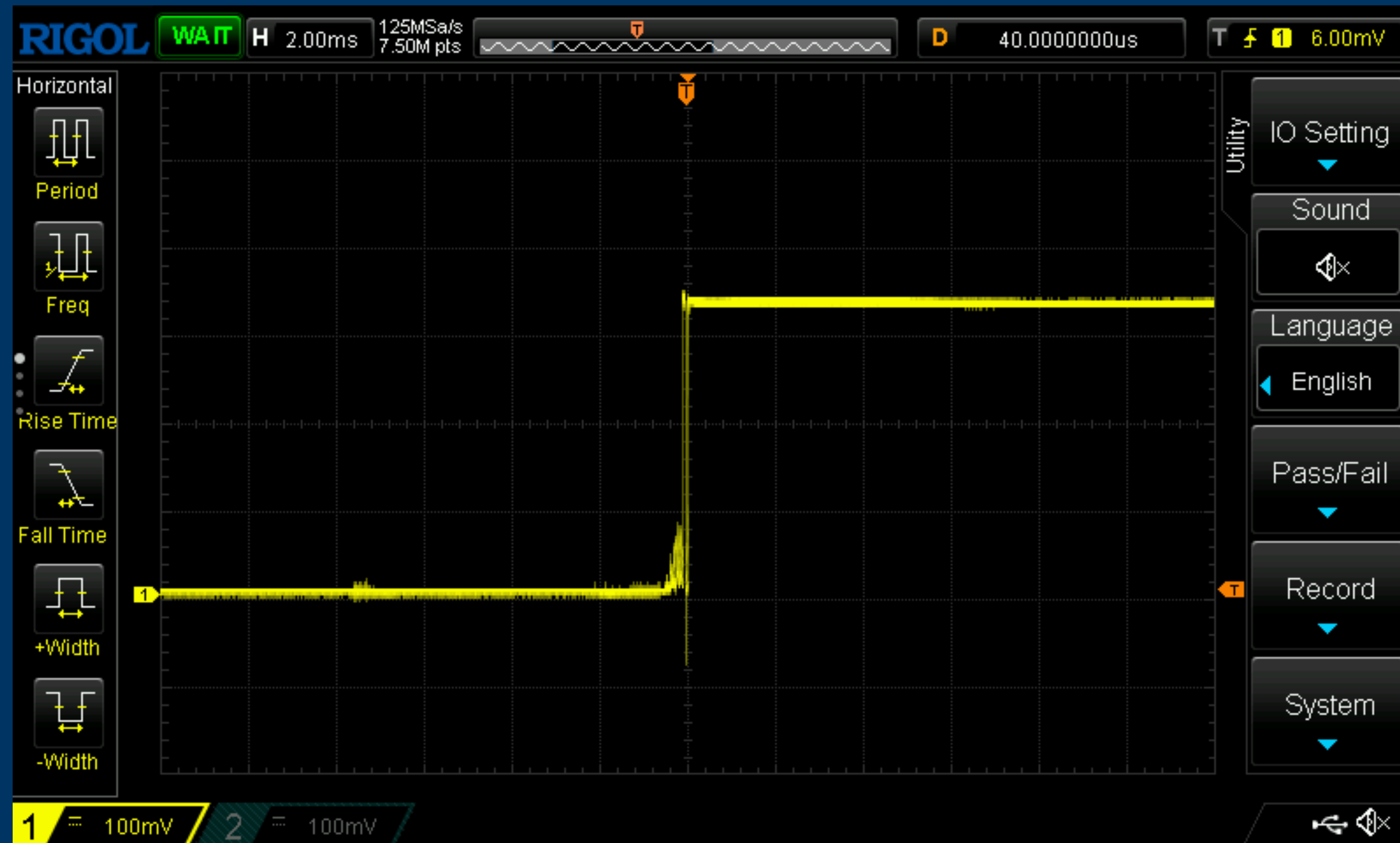
- 7 priority levels
 - level 1 = low priority
 - level 7 = NMI
- Interrupt handling
- ISR constraints

Hands-on

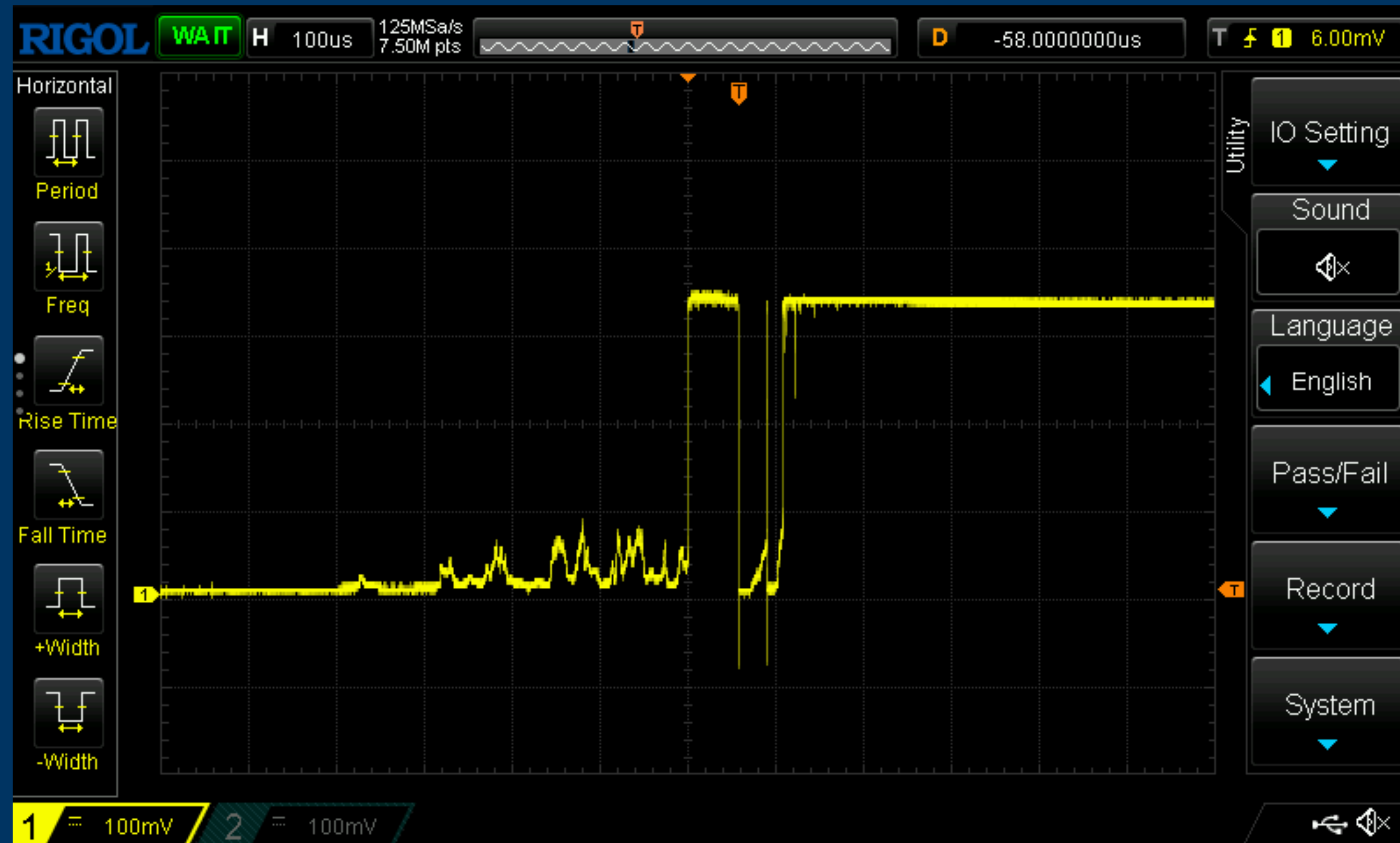
Interrupt-driven callback

- Define InterruptHandlerContext
- Deal with opaque pointer in interrupt handler
- Update Input.init
- Update main
- Test

Bouncing



Bouncing



Debouncing

Debouncing

- Start timer in input interrupt handler
- Ignore input state changes while timer is running
- Check input state when timer expires
- Typical timer duration: 2-20 ms

Hands-on

Debouncing

- Enable ESP Timer ISR dispatch
- Add Timer.swift and Errors.swift, update BridgingHeader.h
- Build
- Create DebouncedInput class
- Update main
- Test

Event Loop

Event Loop

- Event Queue
- Posting from ISR

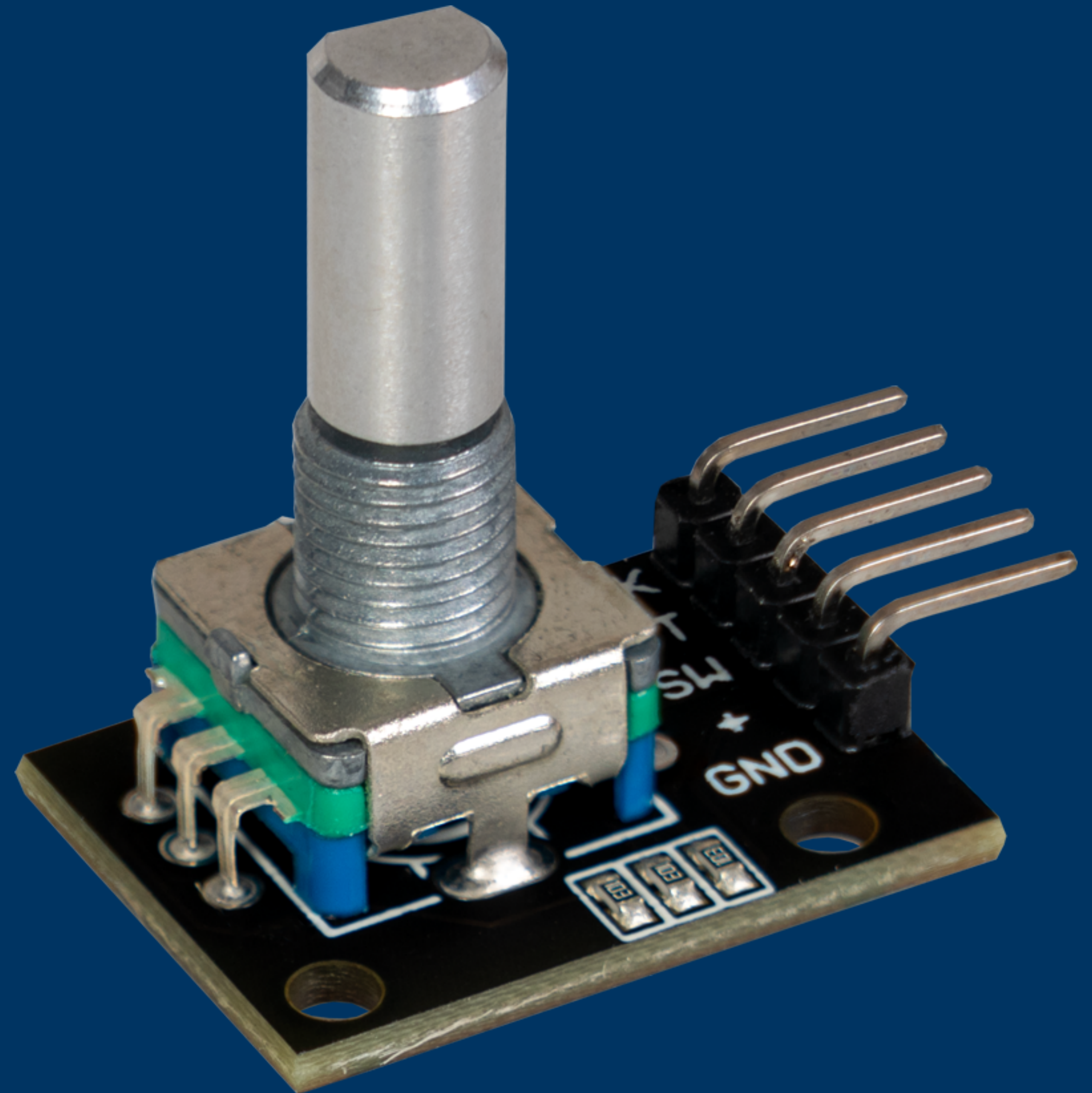
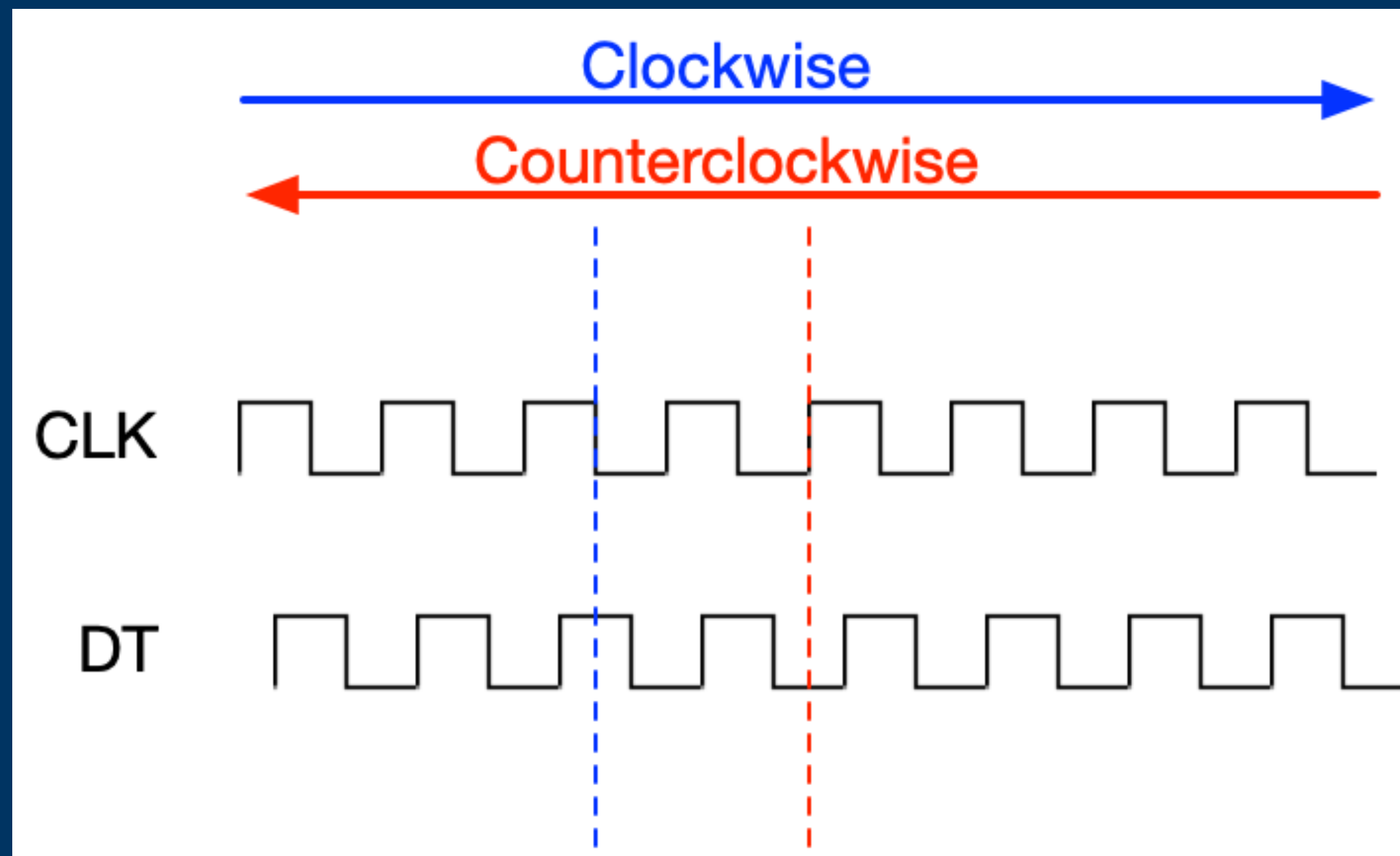
Hands-on

Event Loop

- Add Queue.swift and update BridgingHeader.h
- Create TimedEvent<Event> and EventLoop<Event>
- Update main
- Test

Rotary Encoder

KY-040

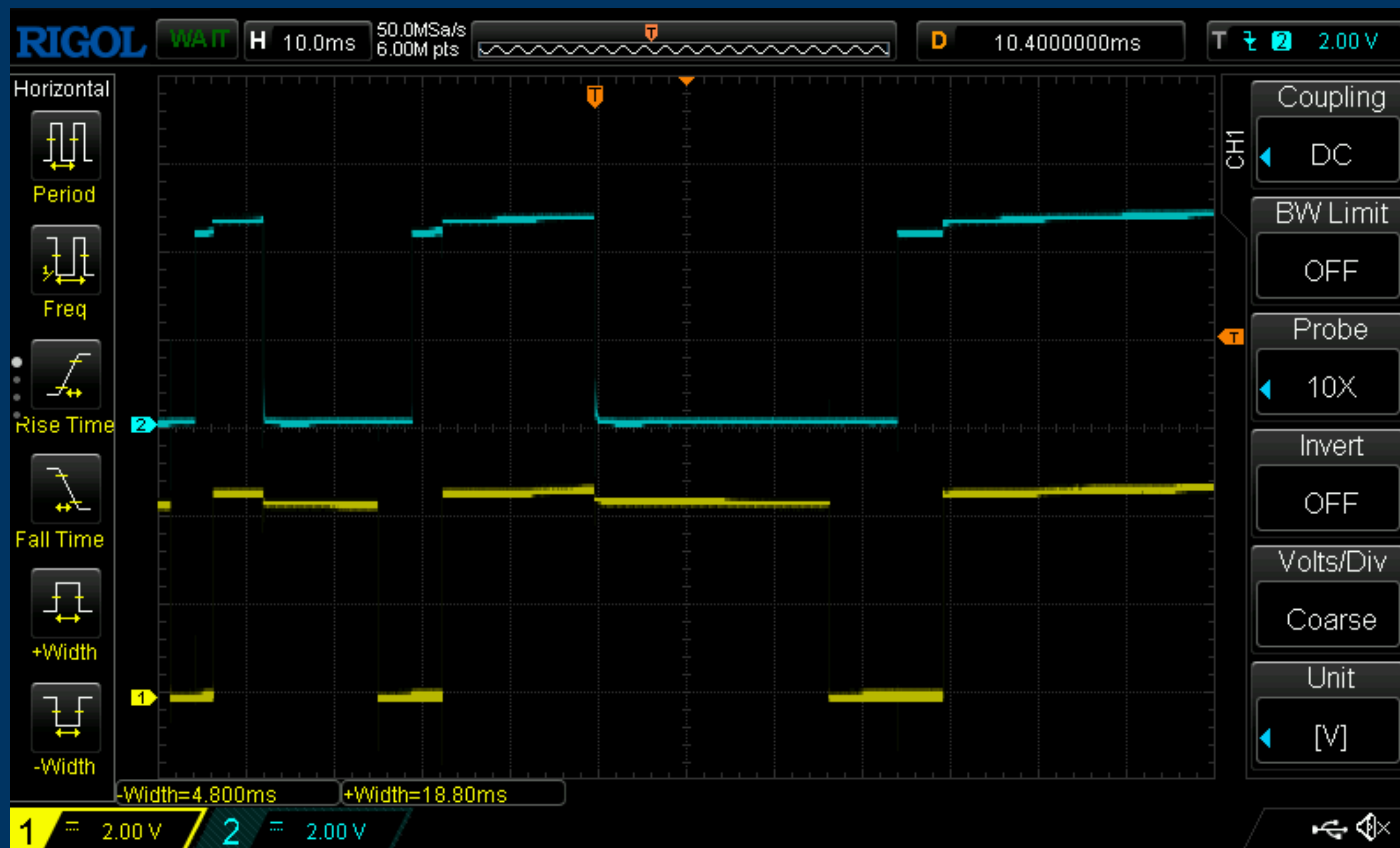


KY-040

Clockwise

CLK

DT

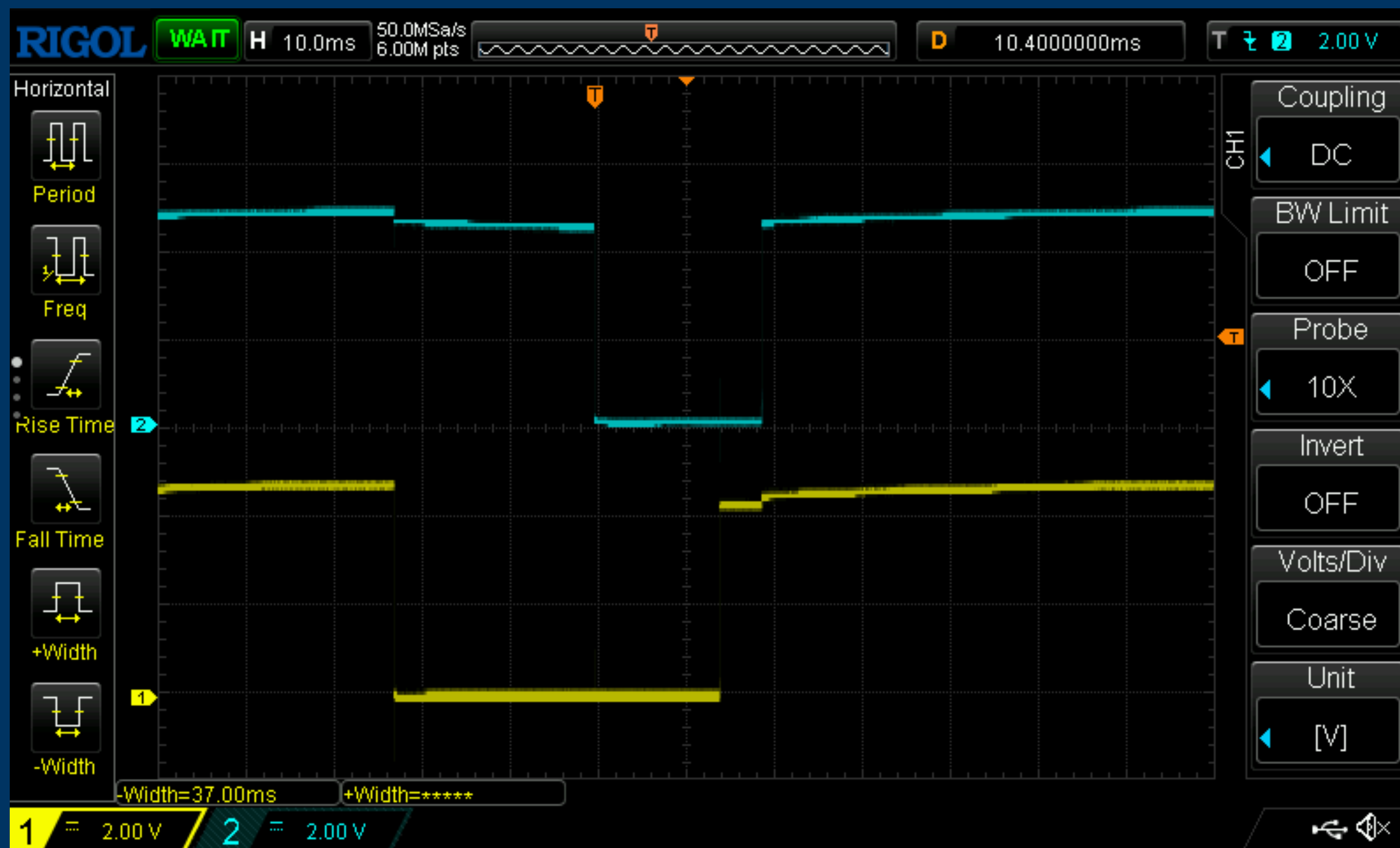


KY-040

Counterclockwise

CLK

DT



Hands-on

Rotary Encoder

- Connect CLK and DT
- Create Direction enum
- Create RotaryController class
- Update main
- Test

Advanced I/O

Beyond binary I/O

Supported by ESP32

- Analog: ADC, PWM
- UART
- I2C, SPI, CAN
- I2S
- RMT

I2C

- Open-drain bus
- Bidirectional
- Master-slave
- SCL, SDA
- Selection: 7-bit identifier
- Typical throughput: 400 kbps

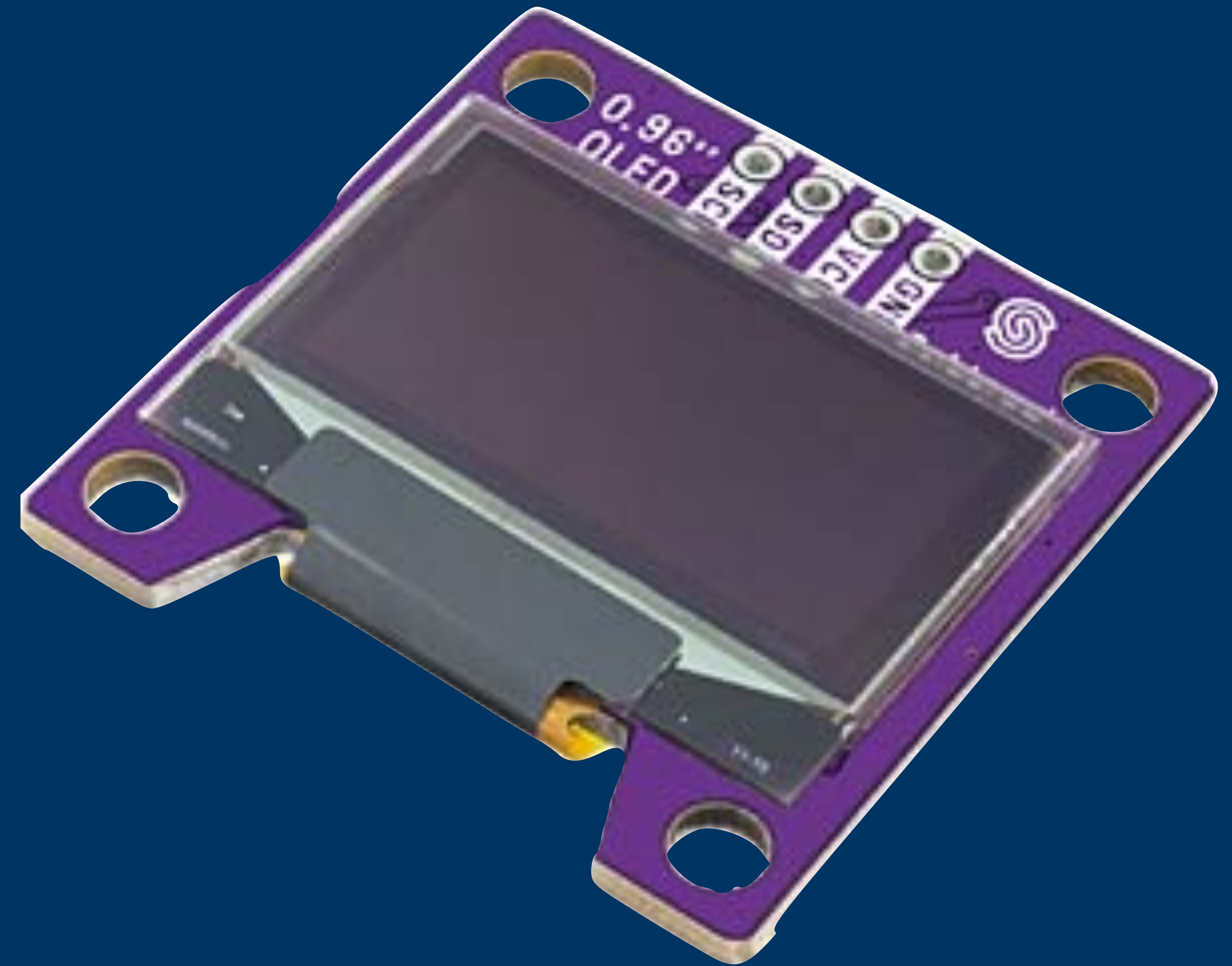
SPI

- Unidirectional
- Master-slave
- Signals: SCLK, MOSI, MISO
- Selection: SS (CS) wire
- Typical throughput > 1 Mbps

Display

SSD1306

- 128x64
- Monochrome OLED
- I2C connection
- 3.3V power



U8g2 Library

- Open source C library
- Lines, shapes
- Text, fonts
- 2 abstraction layers
 - display support: I2C, SPI
 - microcontroller support: requires driver (u8g2-hal-esp-idf)

Hands-on

Display

- Download and add u8g2 and u8g2-hal-esp-idf
- Apply patch to u8g2-hal-esp-idf
- Add Swift wrappers
- Display counter value
- Finish Display implementation
- Draw bargraph

Bluetooth LE

Generic Access Profile (GAP)

No connection (yet)

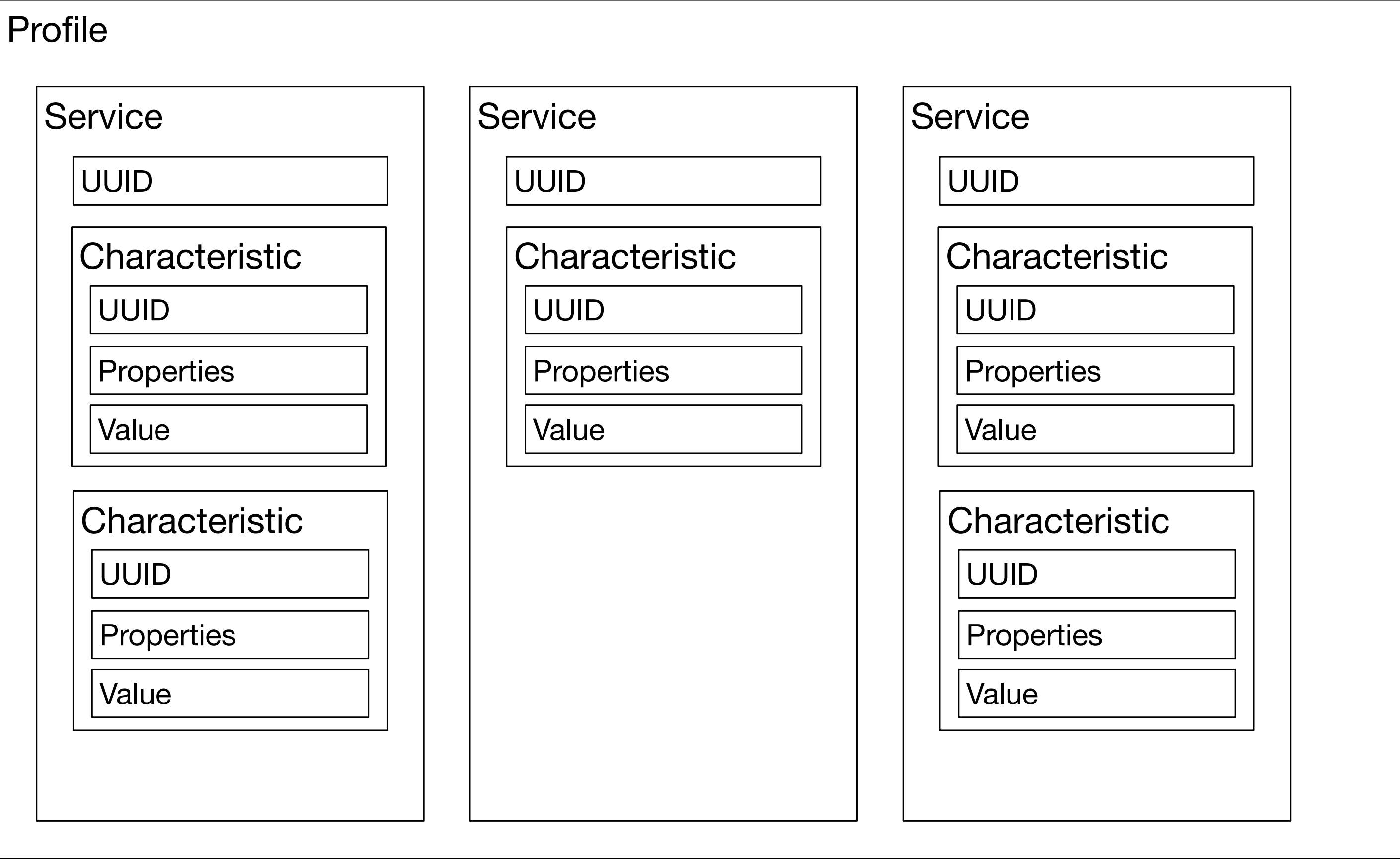
- Broadcasting Roles
 - Broadcaster
 - Observer
- Connecting Roles
 - Central
 - Peripheral

Generic Attribute Profile (GATT)

Connected state

- Device Roles
 - Client
 - Server
- Security
- Device Profile

Device Profile



BLE UUID

- Different lengths
 - 16-bit
 - 32-bit
 - 128-bit
- Well-known services and characteristics
- Central registry ("assigned numbers")

BLE with ESP-IDF

- Bluetooth Stacks
 - NimBLE
 - Bluedroid
- NimBLE Swift wrapper

String support

Advanced String support

- What is "advanced"?
 - Unicode grapheme clusters
 - count, compare (sort), Set/Dictionary, case, split...
- Embedded Swift constraints and workaround
 - libUnicodeDataTables.a
 - utf8 and unicodeScalars views

Hands-on

Bluetooth Peripheral

- Add wrappers
- Implement decoder for UUID128 strings
- Create service and characteristic
- Start advertising
- Test with LightBlue

Core Bluetooth

Core Bluetooth

- Delegate-based
- Main classes
 - CBCentralManager
 - CBPeripheral
 - CBService, CBCharacteristic
- Values: Data
- Permission in Info.plist

BluetoothConnectionManager

- Simple Core Bluetooth wrapper
- Single service, single characteristic
- Not production-ready
- Interface
 - Generic value
 - @Observable: state, readValue
 - write(value:)

Hands-on

iOS Application

- Create SwiftUI App
- Add BluetoothConnectionManager
- Handle state in ContentView
- Test
- Finish BluetoothConnectionManager implementation
- Update ContentView
- Test

AccessorySetupKit

AccessorySetupKit

- Discovery framework
- Bluetooth, Wi-Fi
- Better privacy
- Consistent UI
- Handles pairing process if needed
- Requires iOS 18

ASAccessorySession

Bluetooth Discovery

- activate with closure
 - activated event: list of known accessories
 - accessoryAdded, accessoryRemoved
 - pickerDidDismiss
- showPicker
 - ASPickerDisplayItem, ASDiscoveryDescriptor

Hands-on

AccessorySetupKit

- Update Info.plist
- Create DiscoveryService class
- Handle events
- Implement showPicker
- Update BluetoothServiceManager
- Test