Embedded Swift Workshop

Swift Island 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)

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

Tools & Resources

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

Hands-on

Setup (offline)

- Install from archive
    - Swift Toolchain
    - Visual Studio Code
    - ESP-IDF plugin for VS Code
    - ESP-IDF framework and tools
- Build and run “blink” project

Pulsing LED

Basic GPIO

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

Hands-on

Blinking LED

- Connect LED & resistor
- Open `00-Start-Here` project with VS Code
- Implement main loop
- 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: only for constants
- Swift to C: `@_cdecl("name")`

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>`

Spans

New in Swift 6.2

```
void myFunction(MyType *ptr, int count);
```

```
func myFunction(_ ptr: UnsafeMutablePointer<MyType>?, _ count: Int)
```

Spans

New in Swift 6.2

```
void myFunction(MyType *ptr, int count);

void myFunction(MyType *__counted_by(count) ptr __noescape, int count);
```

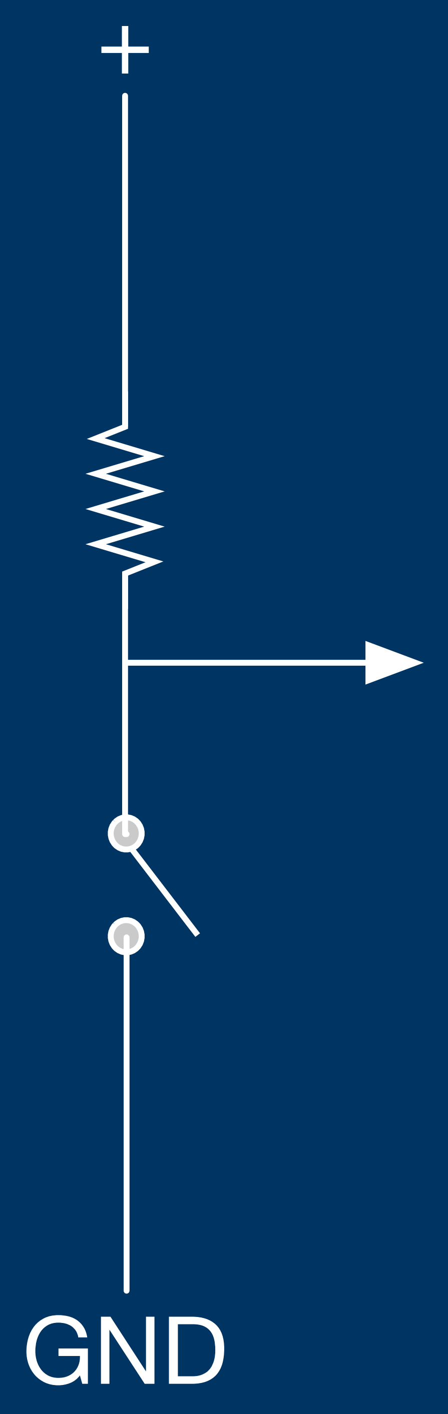
```
func myFunction(_ ptr: UnsafeMutablePointer<MyType>?, _ count: Int)

func myFunction(_ span: Span<MyType>?)
```

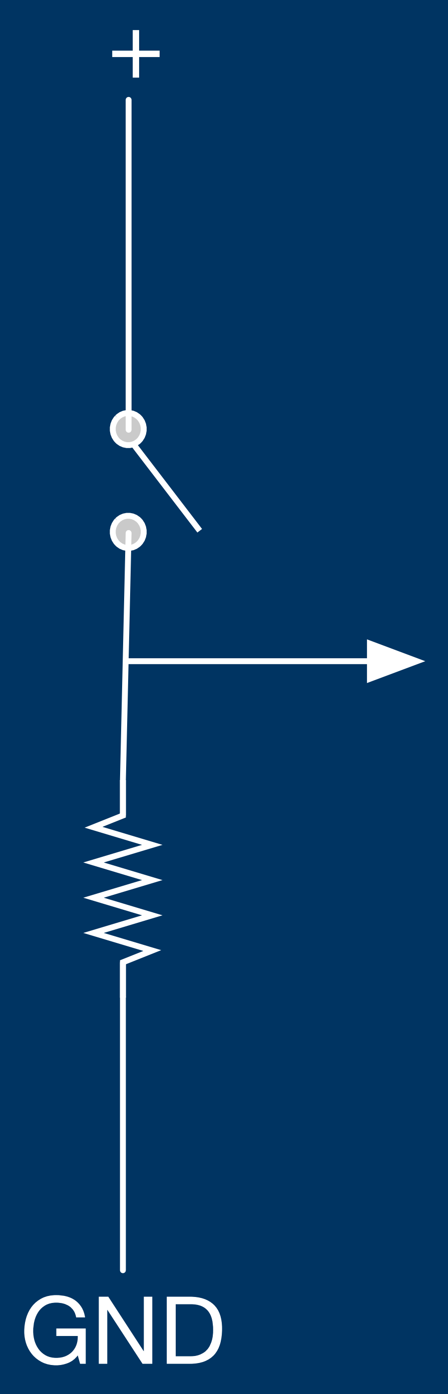
- WWDC 2025: Safely mix C, C++, and Swift

Simple Input

GPIO Input Setup

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





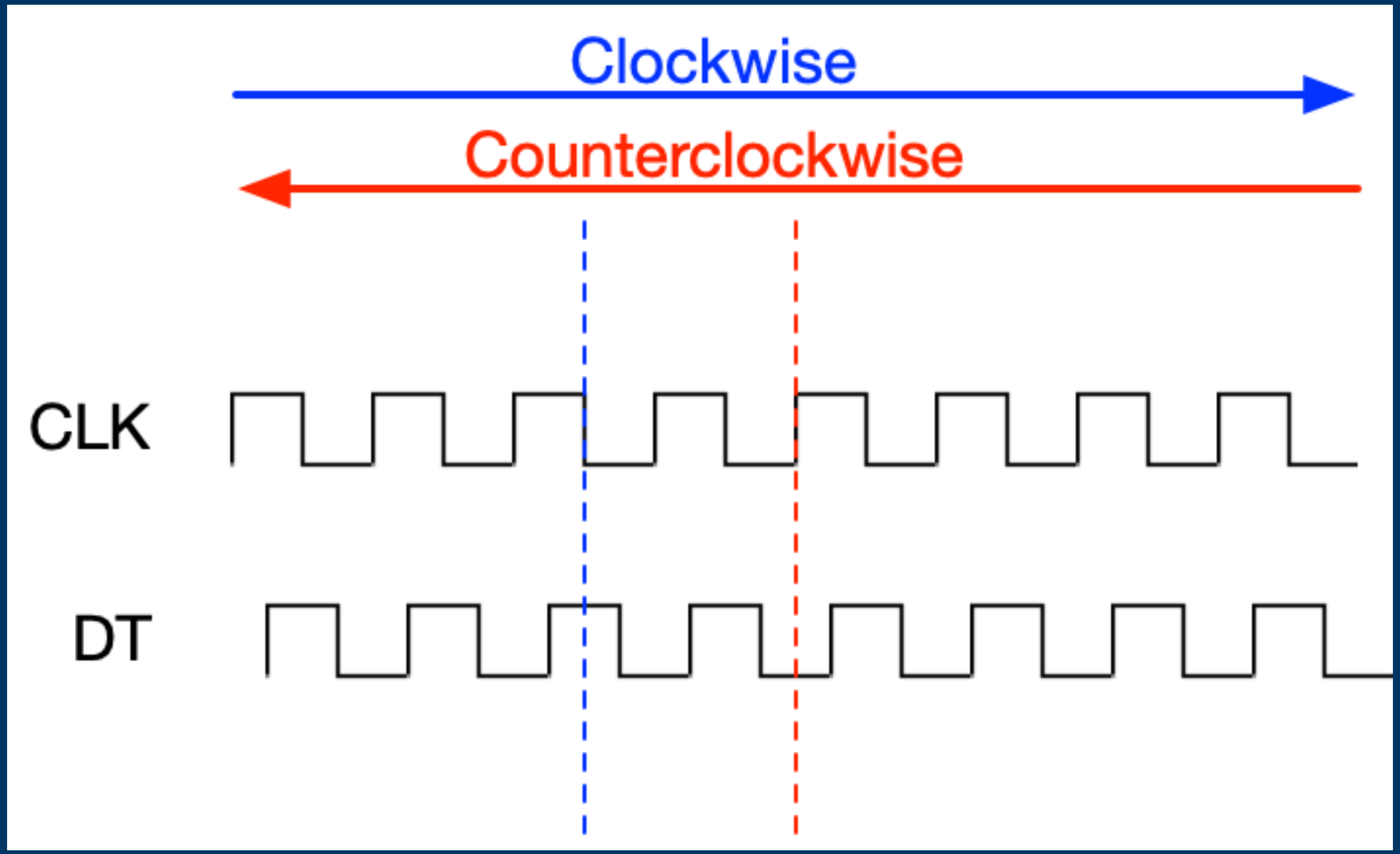
Hands-on

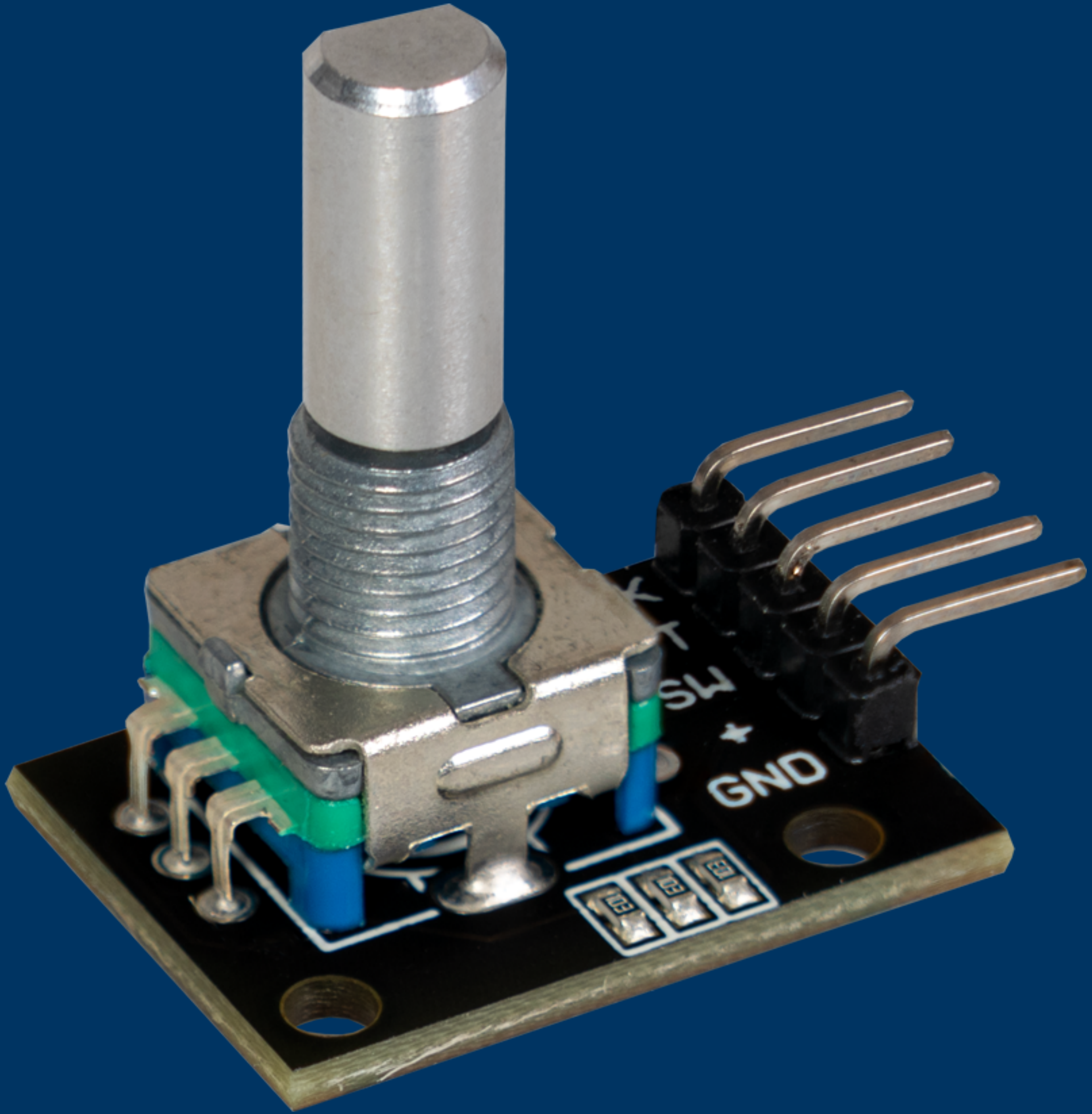
Simple input

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

Rotary Encoder

KY-040



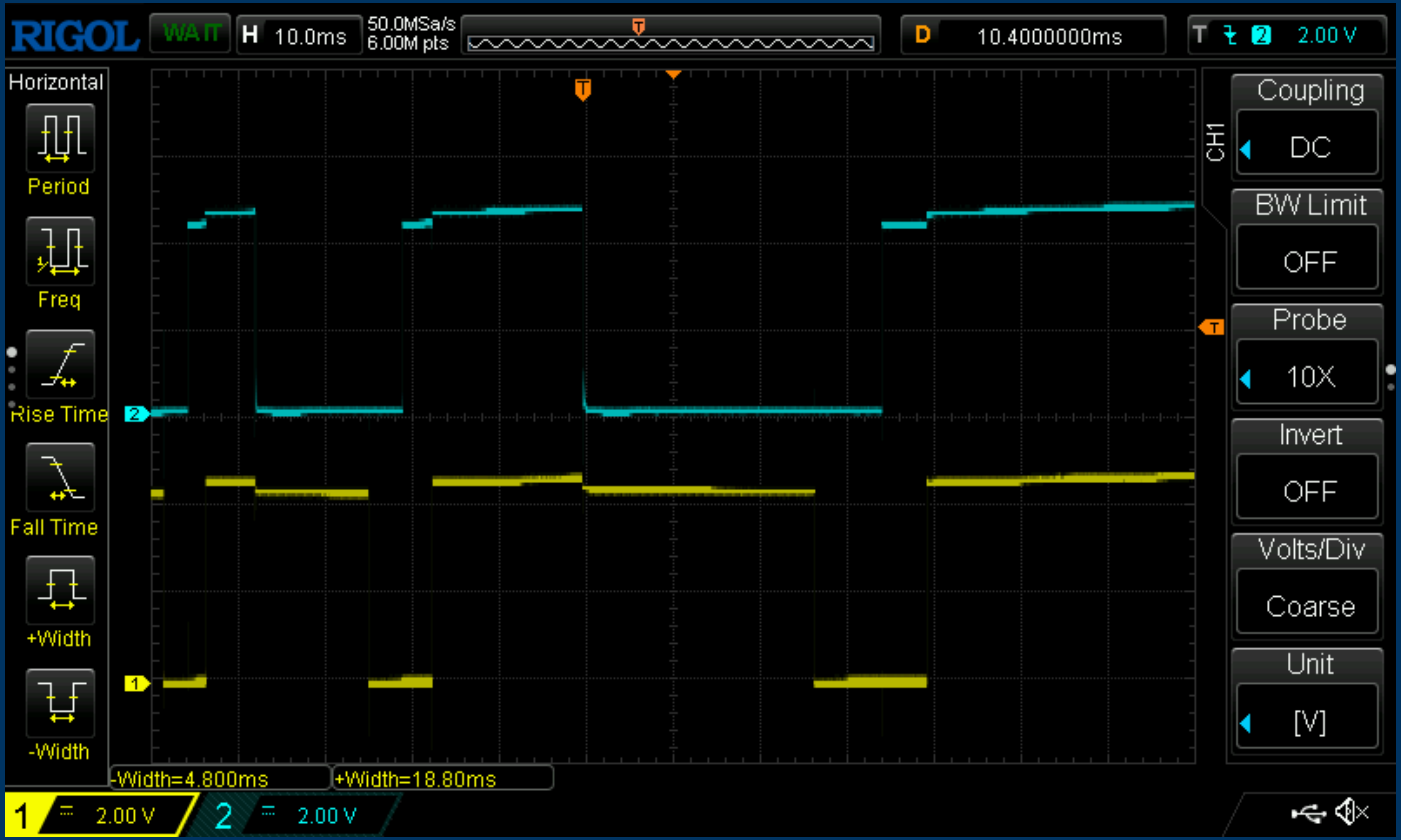


KY-040

Clockwise

CLK

DT



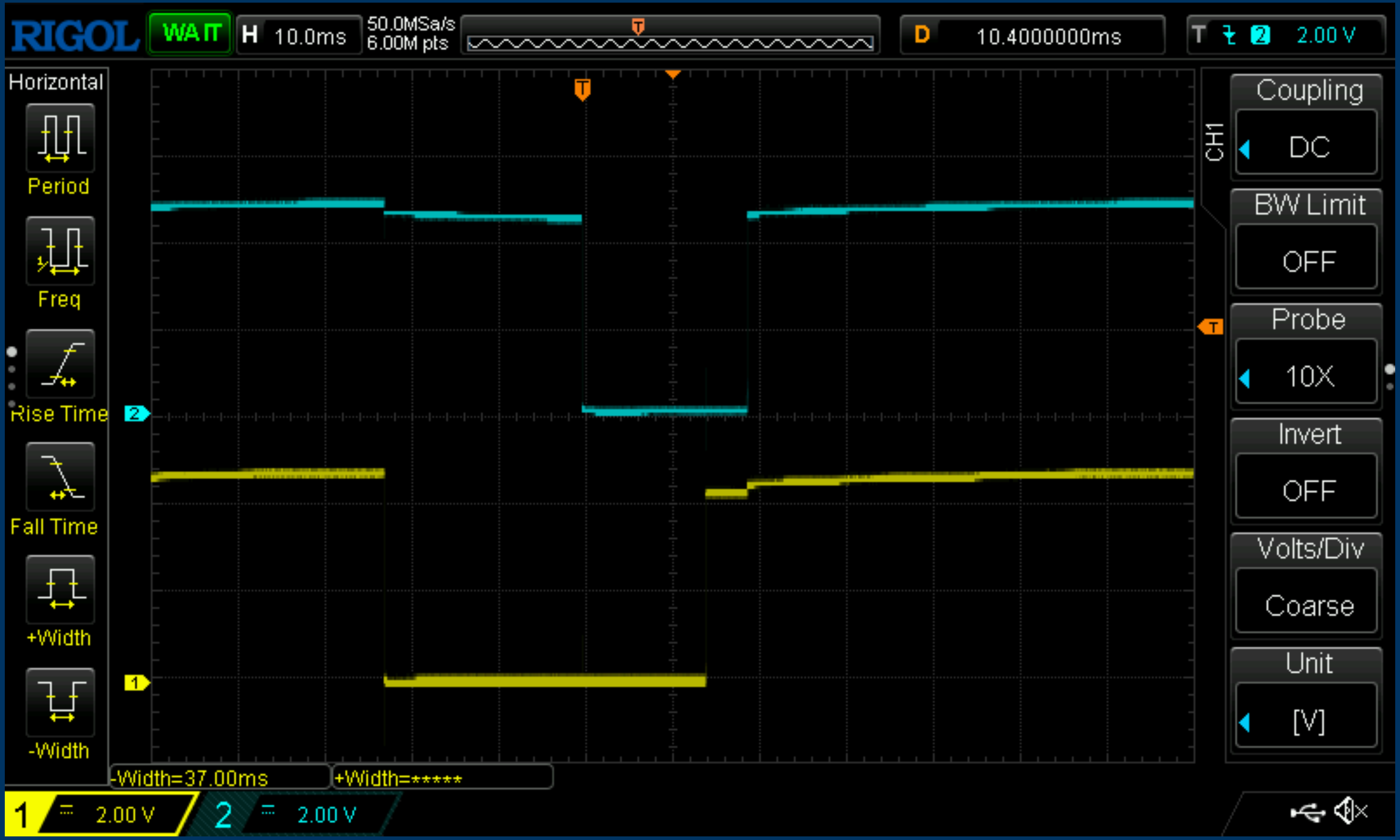
-Width=4.800ms +Width=18.80ms

KY-040

Counterclockwise

CLK

DT



-Width=37.00ms
+Width=*****

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

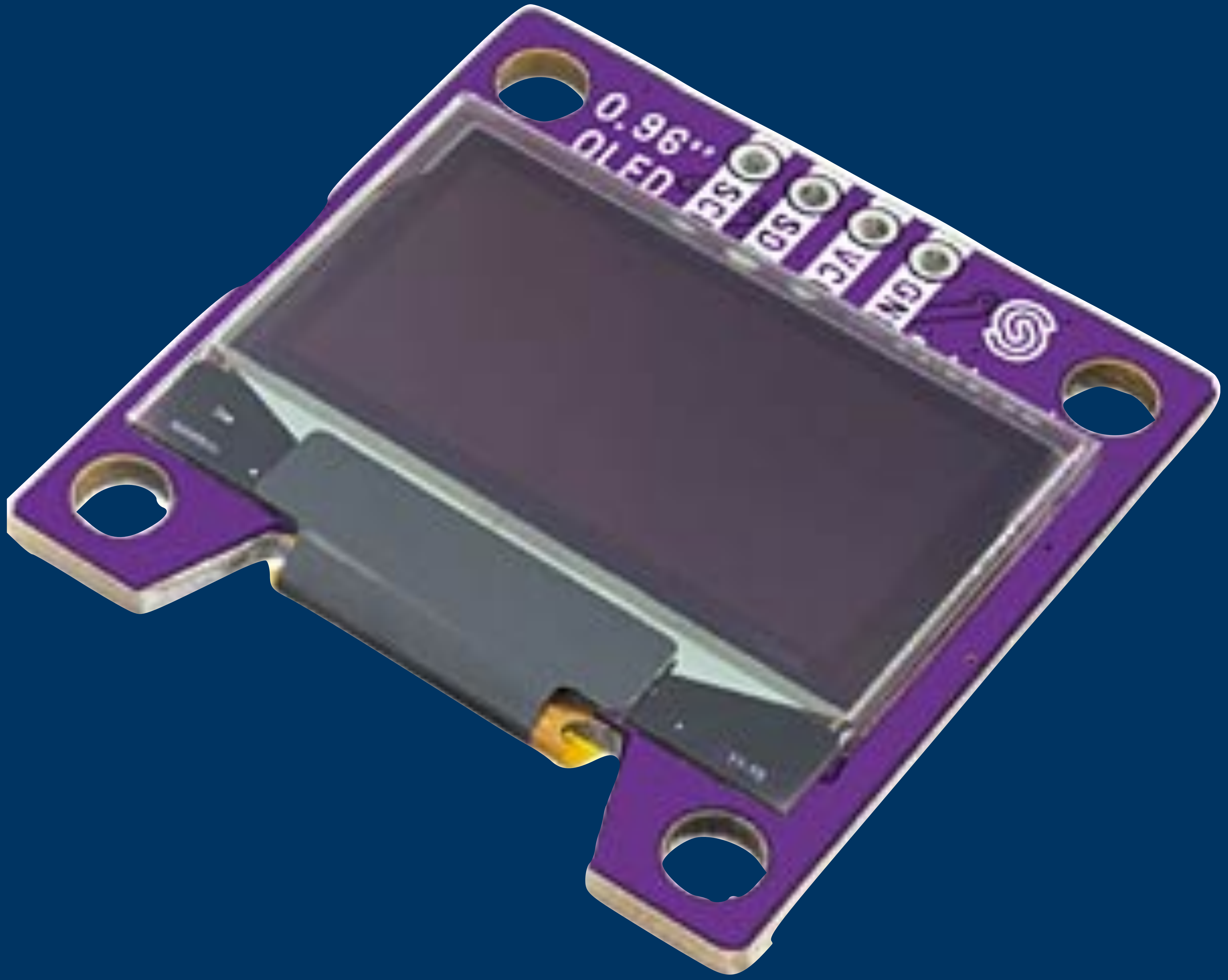
I2C

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

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: drivers included
  - microcontroller support: requires driver (u8g2-hal-esp-idf)

Hands-on

Display

- Add u8g2 and u8g2-hal-esp-idf
- Apply patch to u8g2-hal-esp-idf
- Add Swift wrappers
- Test with text
- Finish Display implementation
- Draw gauge

Images

Images

- No filesystem
  - Link contents of file as binary data
- Storage and memory constraints
  - CCITT G4 algorithm
  - TIFF wrapper
  - Progressive decoding

Hands-on

Images

- Add TIFF_G4 and resources
- Update BridgingHeader
- Expose `swiftIslandLogoPtr()` and `swiftIslandLogoSize()`
- Decode image in `TiffImage.draw()`
- Display image
- Add `inverted` flag
- Handle refresh

Wrap Up