Gophercon, 10 July 2024

...in the smallest of places

Patricio Whittingslow

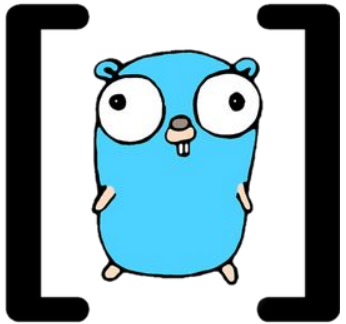
soypat@github

Disclaimer

All views and opinions are my own and do not reflect my employer's point of view.

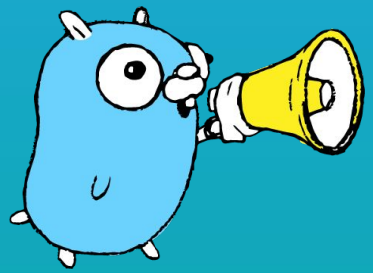
About me

- Mechanical Engineer
- Gopher since 2019
- Embedded systems
- Simulation
  - Rigid body dynamics
  - Solid mechanics (FEM)
- OSS contributor
- Taught Python



github.com/tinygo-org
github.com/gonum

...in the smallest of places



Agenda

- History of Software Engineering
- It's C all the way down
- New languages
- Microcontroller Units (MCUs)
- Live Coding: MCU Registers
- Mapped Memory

History of Software Engineering

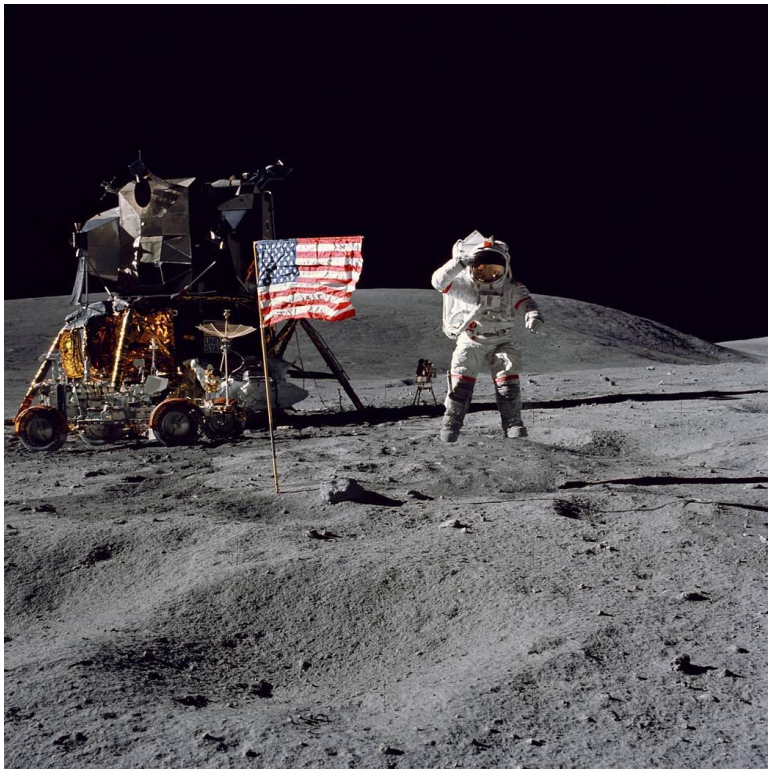
1960-1970

In the beginning…

There was assembly

History of Software Engineering - Apollo Moon Landings

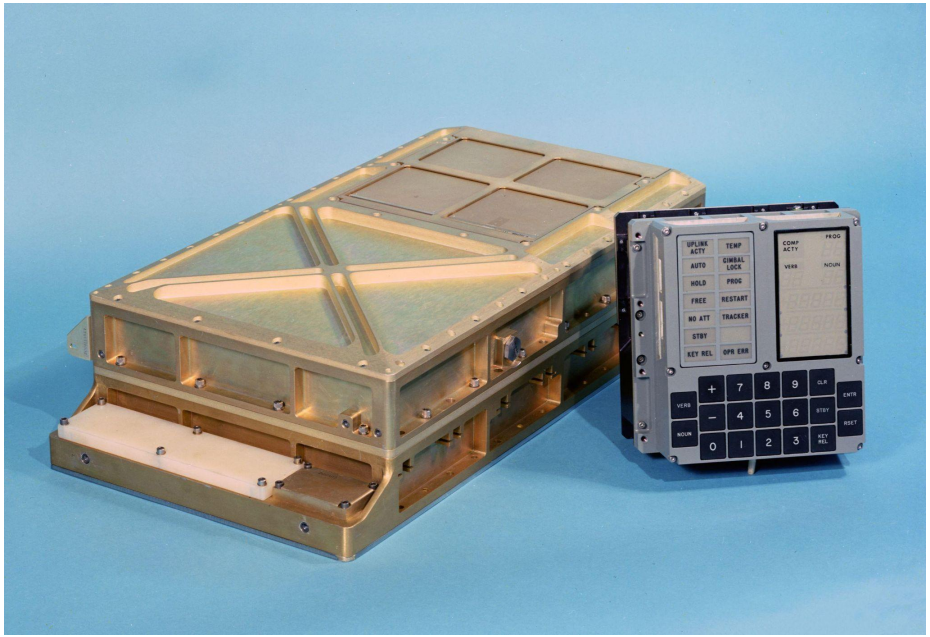
https://www.nasa.gov/image-detail/amf-7029784/



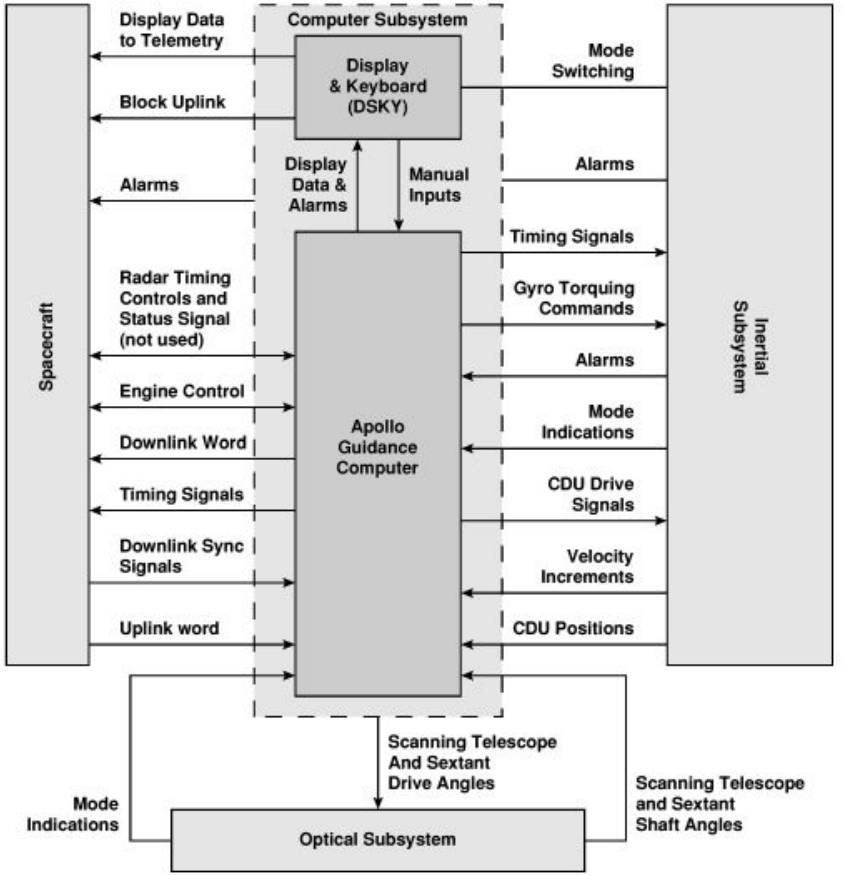
https://www.nasa.gov/image-detail/john-w-youngs-lunar-salute/

History of Software Engineering - Apollo Moon Landings

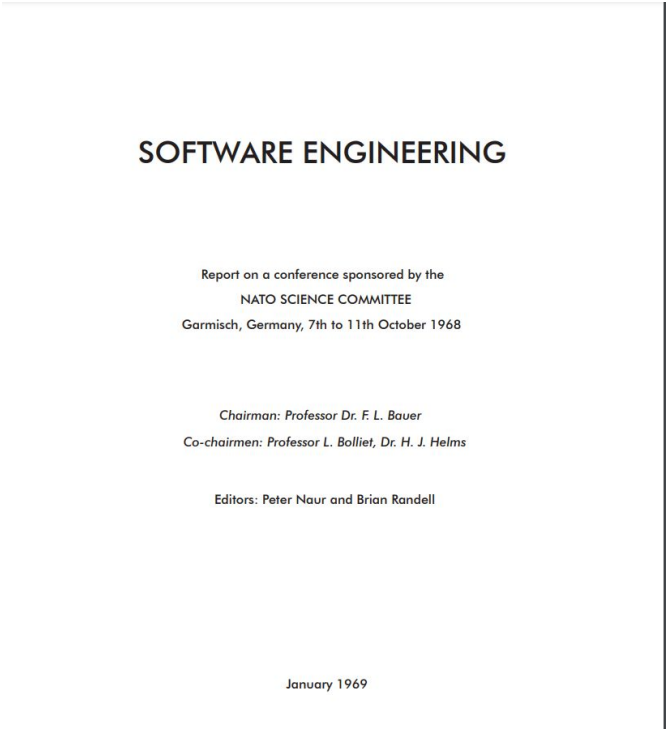
Apollo Guidance Computer (AGC)



https://www.nasa.gov/history/afj/compessay.html



History of Software Engineering - Beginnings



SOFTWARE ENGINEERING

Report on a conference sponsored by the
NATO SCIENCE COMMITTEE
Garmisch, Germany, 7th to 11th October 1968

Chairman: Professor Dr. F. L. Bauer
Co-chairmen: Professor L. Bolliet, Dr. H. J. Helms

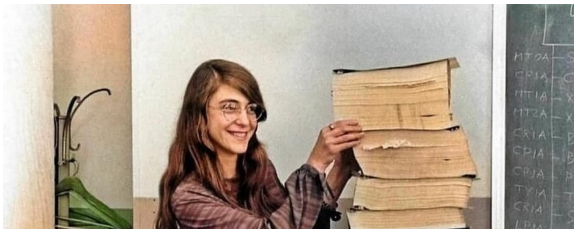
Editors: Peter Naur and Brian Randell

January 1969

https://mitmuseum.mit.edu/collections/object/GCP-00009689

https://www.scrummanager.com/files/nato1968e.pdf

History of Software Engineering - Beginnings



SOFTWARE ENGINEERING

Report on a conference sponsored by the

NATO SCIENCE COMMITTEE

Garmisch, Germany, 7th to 11th October 1968

https://mitmuseum.mit.edu/collections/object/GCP-00009689

https://www.scrummanager.com/files/nato1968e.pdf

History of Software Engineering - Beginnings

- "The design process is an iterative one"
- "Selig's picture requires a feedback loop, for **monitoring** of the system. One must collect data on system performance, for use in future improvements."

```
35                  COUNT*  $$/DAPGT
36
37      # CONTROL REACHES THIS POINT UNDER EITHER OF THE FOLLOWING TWO CONDITIONS ONCE THE DESCENT ENGINE AND THE DIGITAL
38      # AUTOPILOT ARE BOTH ON:
39      #       A) THE TRIM GIMBAL CONTROL LAW WAS ON DURING THE PREVIOUS Q,R-AXIS TIME5 INTERRUPT (OR THE DAPIDLER
40      #          INITIALIZATION WAS SET FOR TRIM GIMBAL CONTROL AND THIS IS THE FIRST PASS), OR
41      #       B) THE Q,R-AXES RCS AUTOPILOT DETERMINED THAT THE VEHICLE WAS ENTERING (OR HAD JUST ENTERED) A COAST
42      #          ZONE WITH A SMALL OFFSET ANGULAR ACCELERATION.
43      # GTS IS THE ENTRY TO THE GIMBAL TRIM SYSTEM FOR CONTROLLING ATTITUDE ERRORS AND RATES AS WELL AS ACCELERATIONS.
44
45      GTS             CAF     NEGONE          # MAKE THE NEXT PASS THROUGH THE DAP BE
46                      TS      COTROLER        #   THROUGH RCS CONTROL,
47                      CAF     FOUR            #   AND ENSURE THAT IT IS NOT A SKIP.
48                      TS      SKIPU
49                      TS      SKIPV
50
51                      CAF     TWO
52                      TS      INGTS           # SET INDICATOR OF GTS CONTROL POSITIVE.
53                      TS      QGIMTIMR        # SET TIMERS TO 200 MSEC TO AVOID BOTH
54                      TS      RGIMTIMR        # RUNAWAY AND INTERFERENCE BY NULLING.
55
56      # THE DRIVE SETTING ALGORITHM
57      #
58      # DEL = SGN(OMEGA + ALPHA*ABS(ALPHA)/(2*K))
59      #                                       2             1/2                 2       3/2
60      # NEGUSUM = ERROR*K + ALPHA*(DEL*OMEGA + ALPHA /(3*K)) + DEL*K   (DEL*OMEGA + ALPHA /(2*K))
61      #
62      # DRIVE = -SGN(NEGUSUM)
63
64                      CA      SR              # SAVE THE SR.  SHIFT IT LEFT TO CORRECT
65                      AD      A               # FOR THE RIGHT SHIFT DUE TO EDITING.
66                      TS      SAVESR
67
68      GTSGO+ON        CAF     TWO             # SET INDEXER FOR R-AXIS CALCULATIONS.
69                      TCF     GOQTRIMG +1
70
71      GOQTRIMG        CAF     ZERO            # SET INDEXER FOR Q-AXIS CALCULATIONS
```

Apollo-11 / Luminary099 / TRIM_GIMBAL_CONTROL_SYSTEM.agc

chrislgarry / Apollo-11

Type / to search

Code | Issues 59 | Pull requests 74 | Actions | Projects 1 | Wiki | Security | Insights

Apollo-11

Public

Watch 1.3k | Fork 6.8k | Star 56.7k

master | 2 Branches | 0 Tags | Go to file t | Add file | Code

wopian chore: fix typo in luminary099 readme ✕ 9c5454c · 3 weeks ago | 547 Commits

.github	ci(markdownlint): upgrade actions/checkout to v3 (#877)	2 years ago
Comanche055	proof: VNP00H to VNPOOH	4 years ago
Luminary099	chore: fix typo in luminary099 readme	3 weeks ago
.editorconfig	Changed charset in .editorconfig (#681)	4 years ago
.gitignore	chore: ignore yaYUL executables	4 years ago
.mdlrc	ci: ignore MD007	2 years ago
CONTRIBUTING.ar.md	Add Ukrainian README and CONTRIBUTING (#903)	8 months ago
CONTRIBUTING.cz.md	Add Ukrainian README and CONTRIBUTING (#903)	8 months ago
CONTRIBUTING.da.md	Add Ukrainian README and CONTRIBUTING (#903)	8 months ago
CONTRIBUTING.de.md	Add Ukrainian README and CONTRIBUTING (#903)	8 months ago
CONTRIBUTING.es.md	Add Ukrainian README and CONTRIBUTING (#903)	8 months ago
CONTRIBUTING.fr.md	Add Ukrainian README and CONTRIBUTING (#903)	8 months ago
CONTRIBUTING.gr.md	Add Ukrainian README and CONTRIBUTING (#903)	8 months ago

About

Original Apollo 11 Guidance Computer (AGC) source code for the command and lunar modules.

apollo | nasa | hacktoberfest | agc

Readme
View license
Activity
56.7k stars
1.3k watching
6.8k forks
Report repository

Releases

No releases published

Packages

No packages published

Contributors 188

```
69                  TCF     GOQTRIMG +1
70
71  GOQTRIMG        CAF     ZERO            # SET INDEXER FOR Q-AXIS CALCULATIONS
```

History of Software Engineering

1970s-Today

History of Software Engineering - Enter C

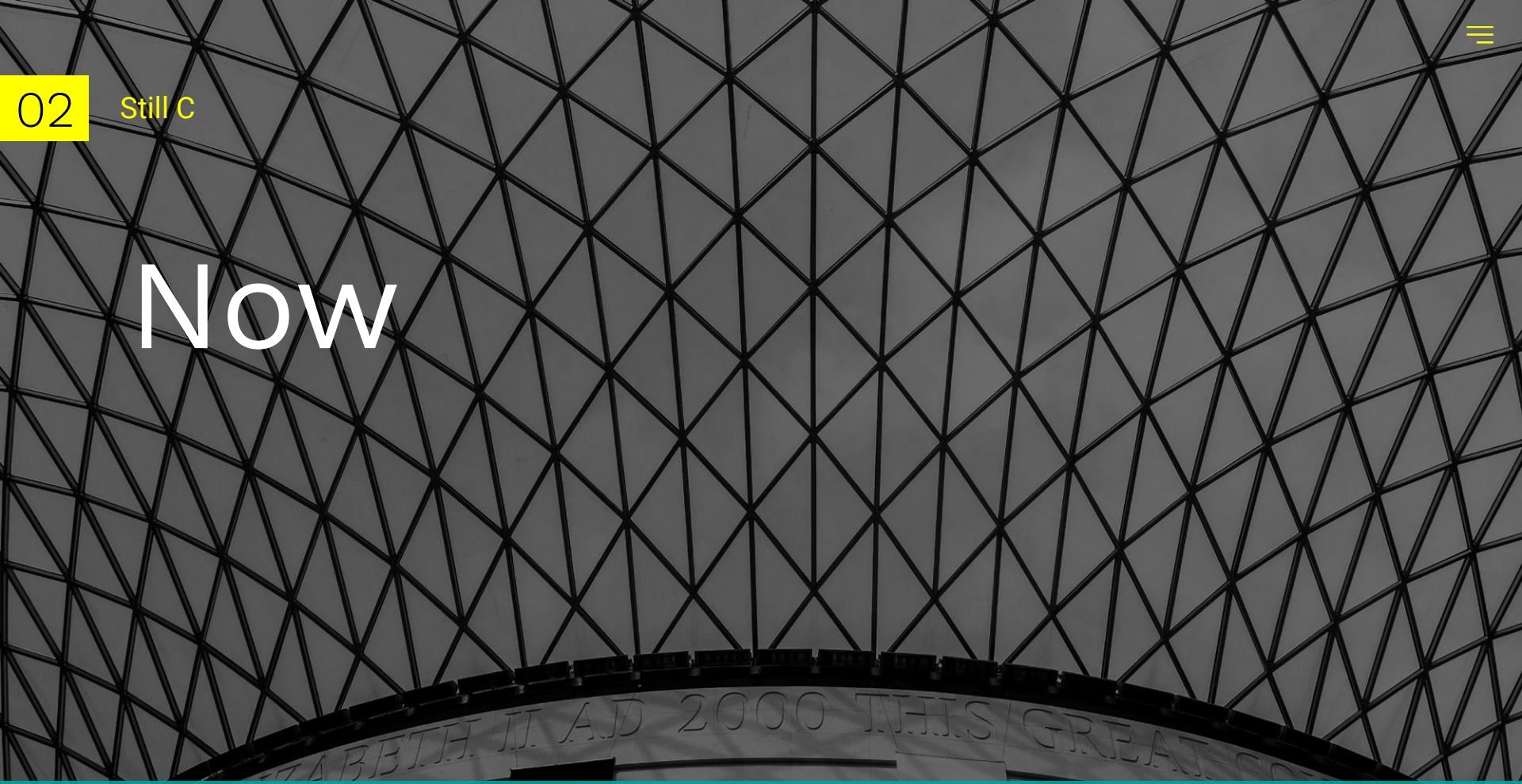
- * -> Algol -> BCPL -> B -> C
- Same mental model as assembly
- Abstractions that map to hardware
- Uncompromising "high" level language

History of Software Engineering - Enter C

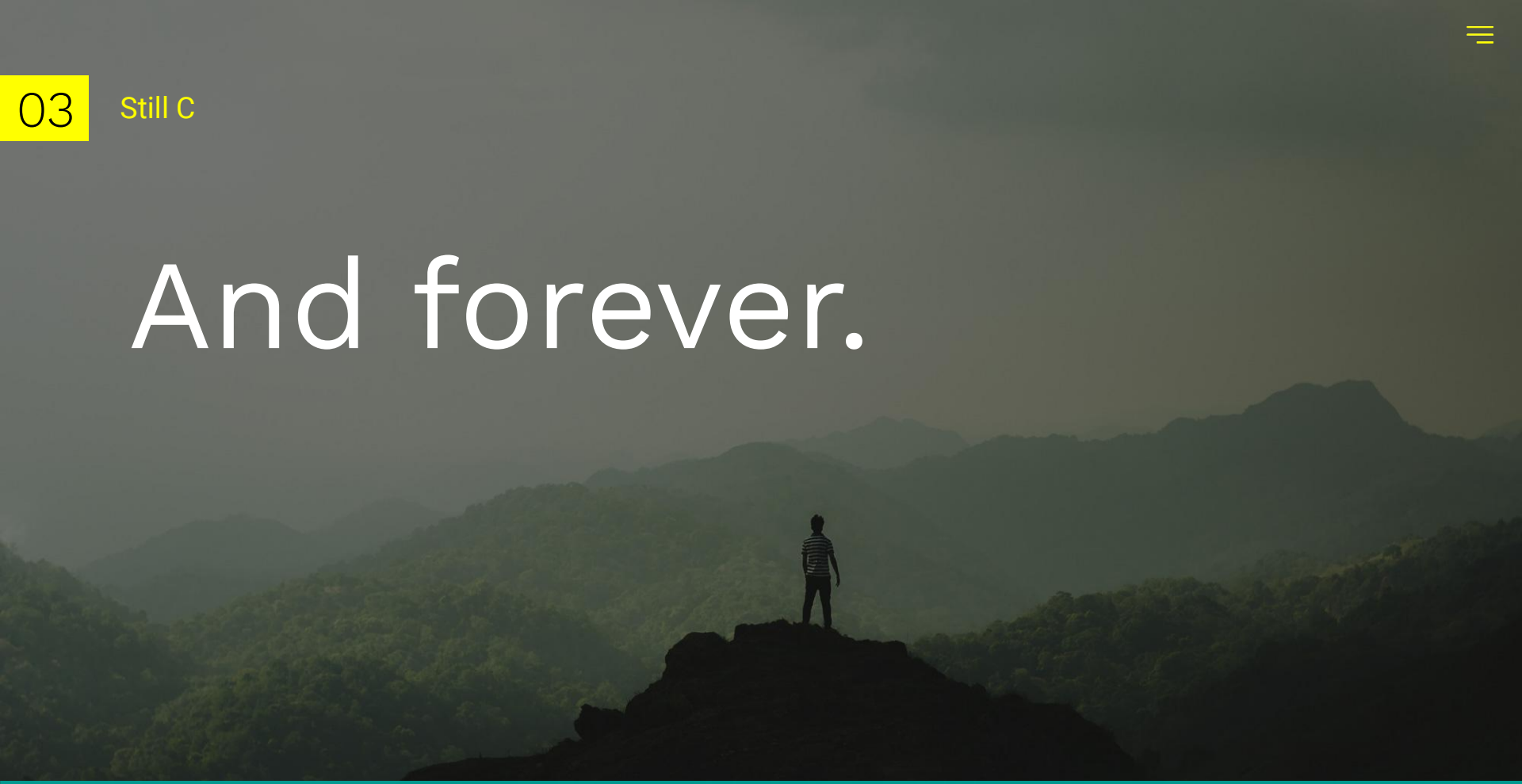
```
int add(int a, int b) {
   return a+b;
}

void main() {
   int sum = add(20, 30);
   printf("%d\n", sum);
}
```

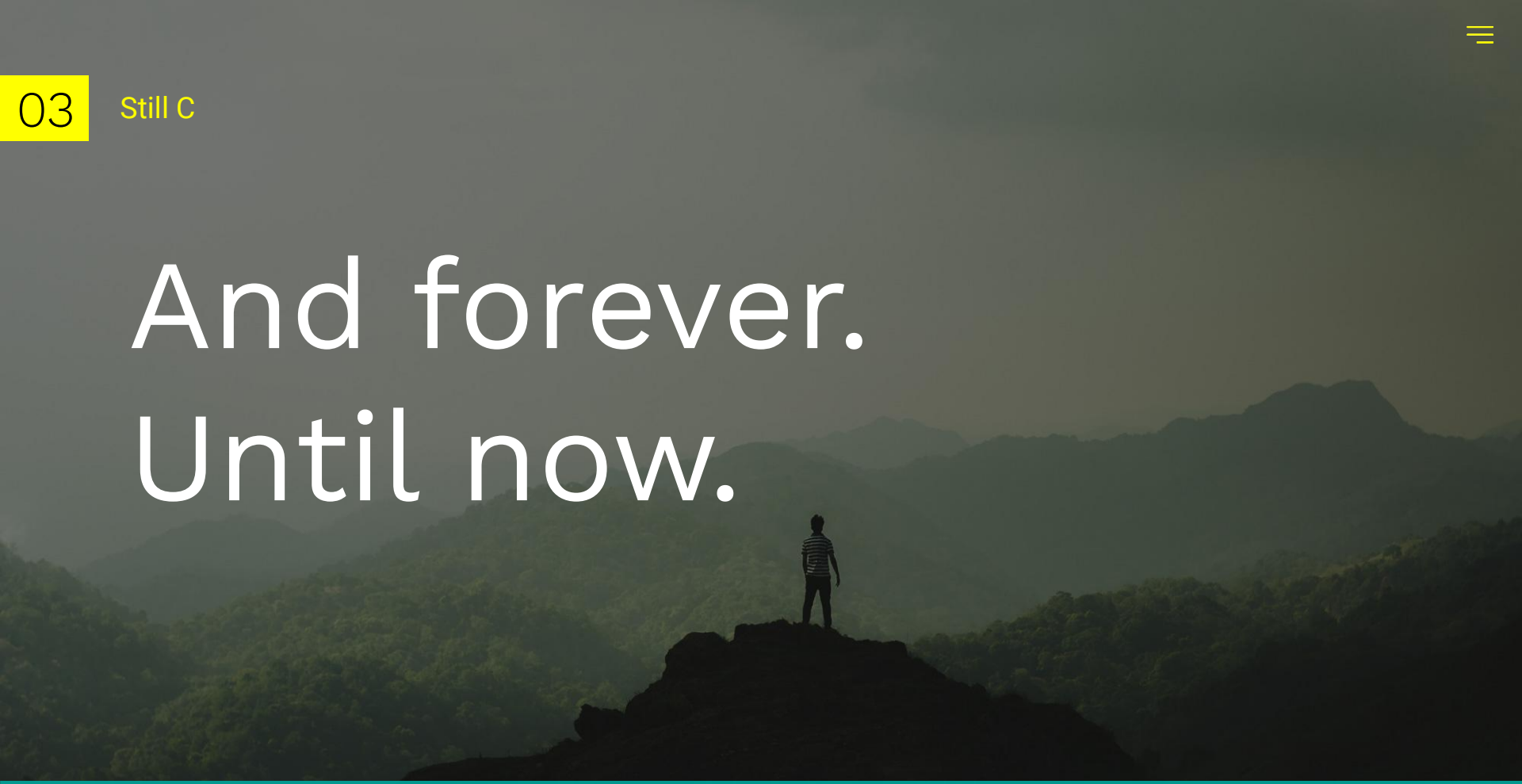
Still C

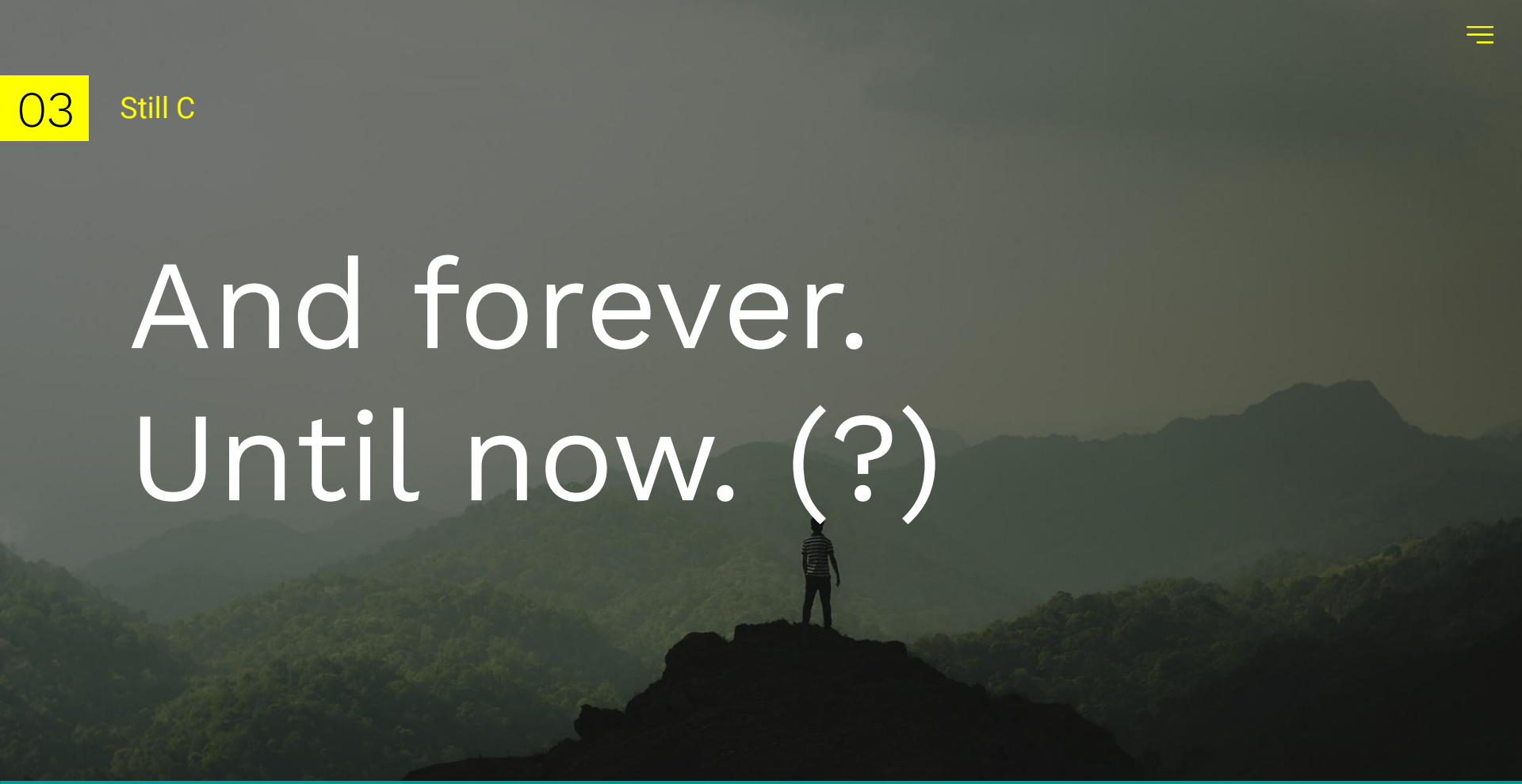
Now



And forever.




And forever.
Until now.

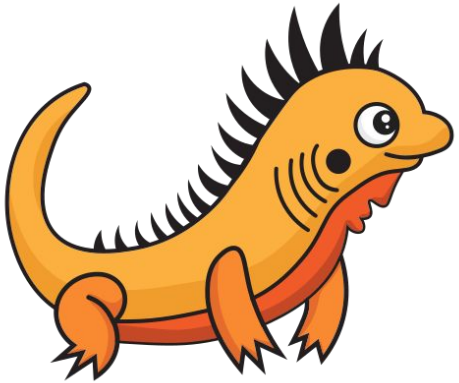


And forever.
Until now. (?)

History of Software Engineering

2010s-Today

- Comptime
- Allocators
- C/C++ compiler
- Future LLVM replacement?



Crablang

- Strong memory guarantees
- Immutability by default
- Established in industry

Go

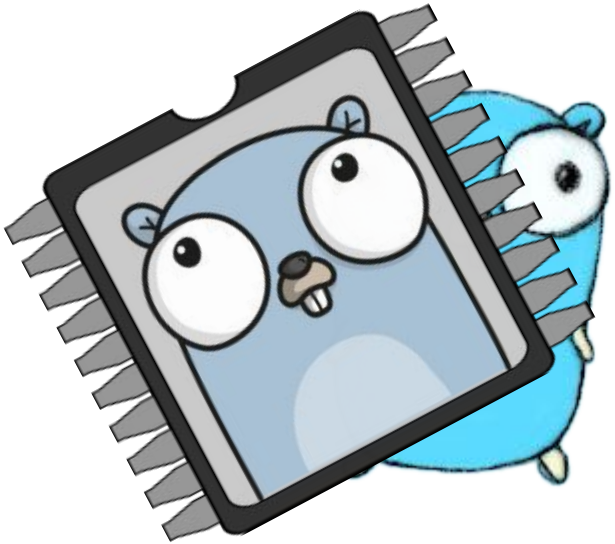
Go

- It's OK
- C with a runtime

Go

- It's OK
- C with a runtime

```
$ tinygo help
  -scheduler string
        which scheduler to use
        (none, tasks, asyncify)
  -gc string
        garbage collector to use
        (none, leaking, conservative)
```



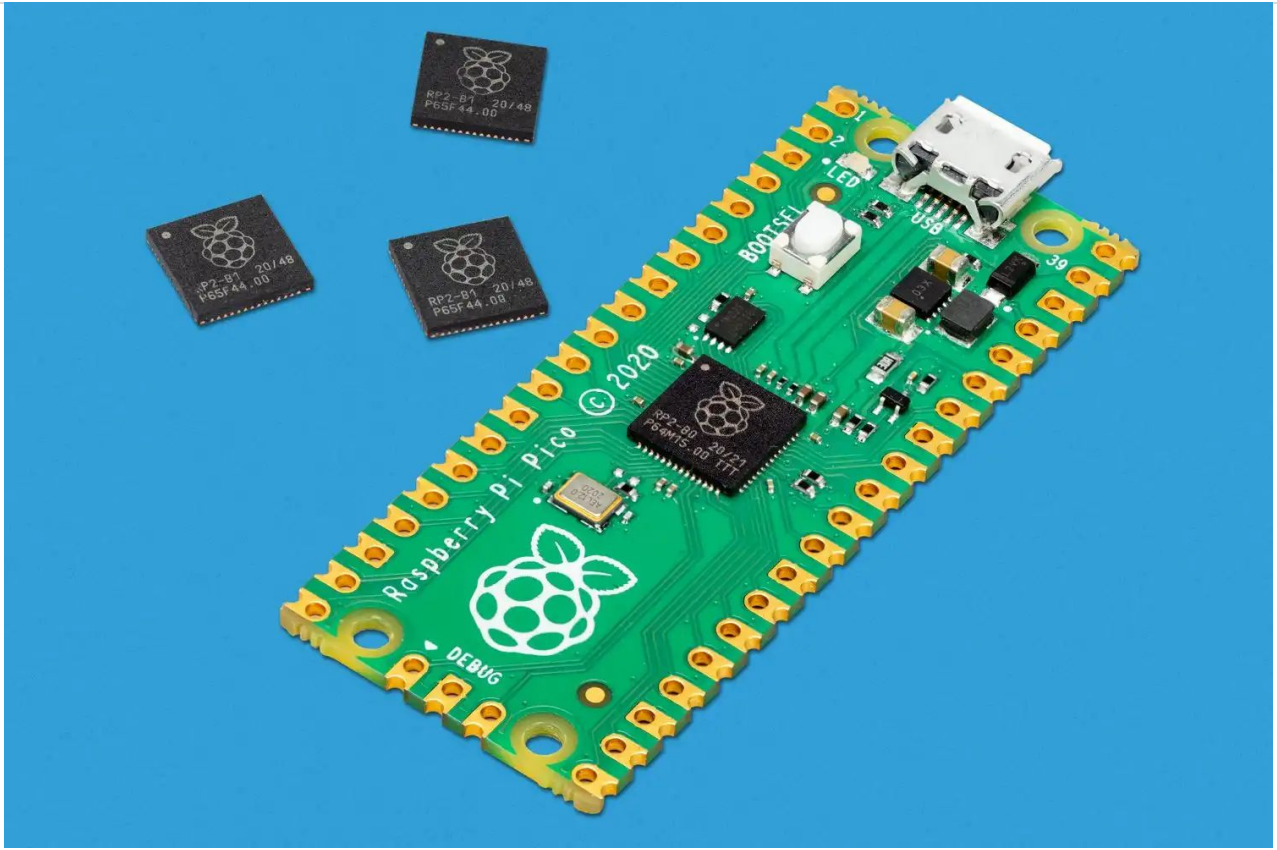
History of Software Engineering - Key Takeaways

- Big software projects in Assembly
  - Assembly -> C
- C -> ???
- New languages viable for MCU control

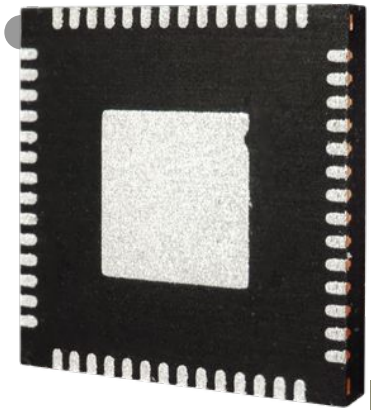
Intro to Embedded Systems

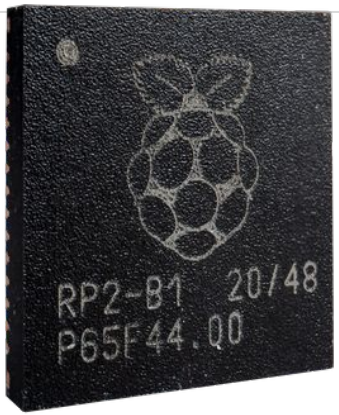
Microcontroller Units (MCUs)

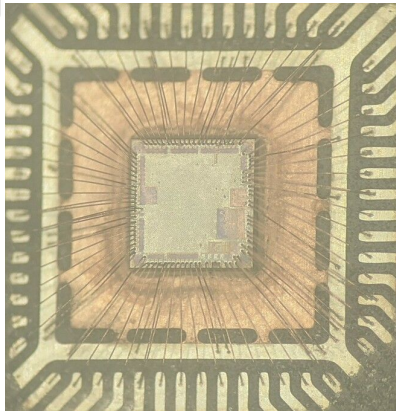
What is a MCU?

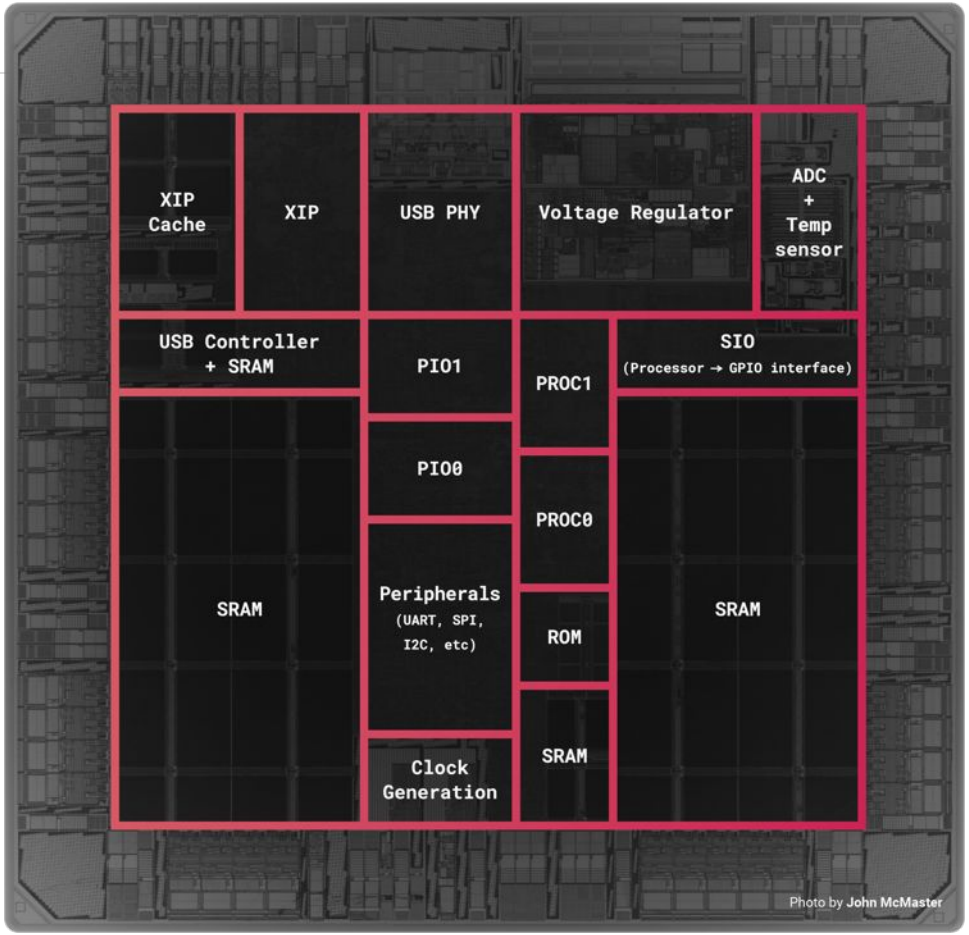


What is a MCU?



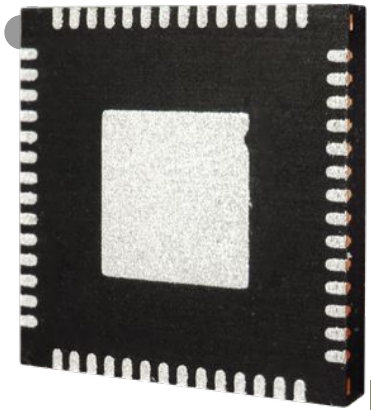


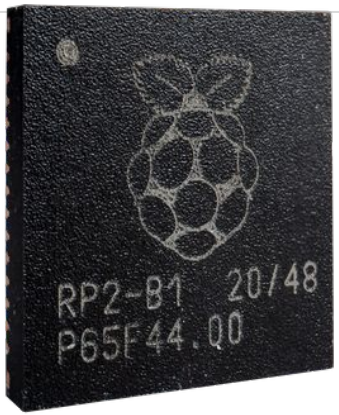


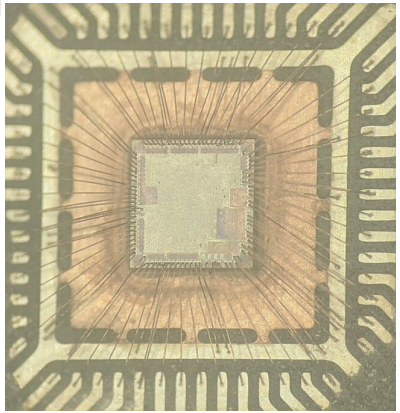


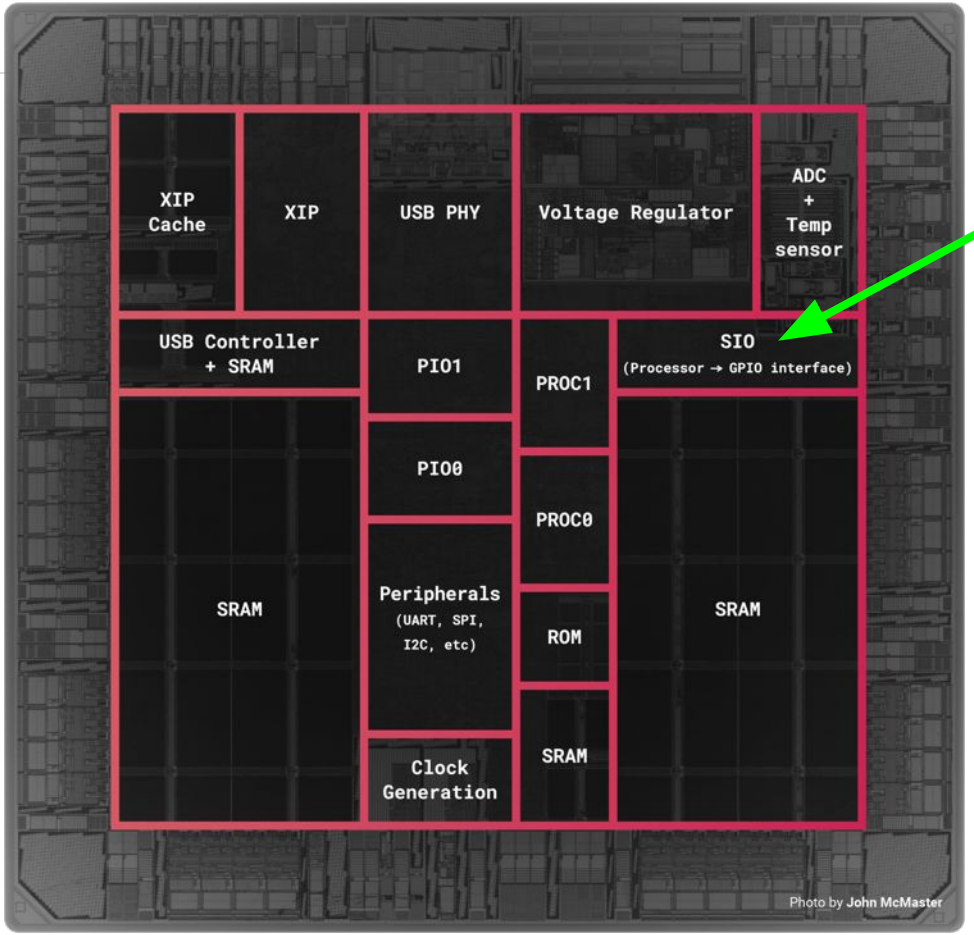
https://twitter.com/johndmcmaster/status/1355092011829719046

What is a MCU?



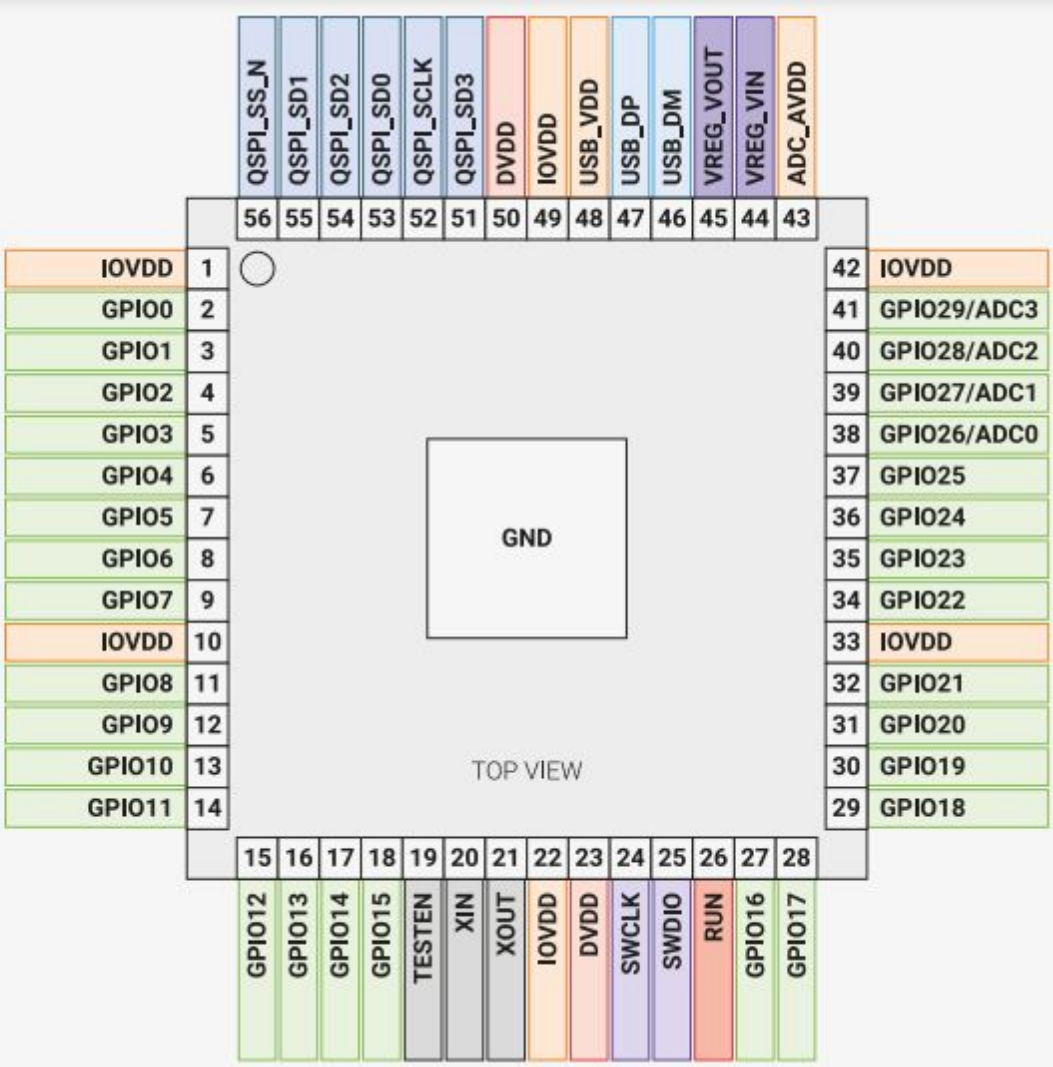






https://twitter.com/johndmcmaster/status/1355092011829719046

7×7mm (reduced ePad size)



2.3. Processor subsystem

The RP2040 processor subsystem consists of two Arm Cortex-M0+ processors — each with its standard internal Arm CPU peripherals — alongside external peripherals for GPIO access and inter-core communication. Details of the Arm Cortex-M0+ processors, including the specific feature configuration used on RP2040, can be found in Section 2.4.

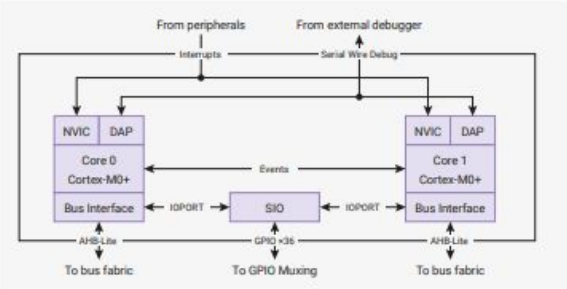


Figure 6. Two Cortex-M0+ processors, each with a dedicated 32-bit AHB-Lite bus port, for code fetch, loads and stores. The SIO is connected to the single-cycle IOPORT bus of each processor, and provides GPIO access, two-way communications, and other core-local peripherals. Both processors can be debugged via a single multi-drop Serial Wire Debug bus. 26 interrupts (plus NMI) are routed to the NVIC and WIC on each processor.

NOTE

The terms *core0* and *core1*, *proc0* and *proc1* are used interchangeably in RP2040's registers and documentation to refer to processor 0, and processor 1 respectively.

The processors use a number of interfaces to communicate with the rest of the system:

- Each processor uses its own independent 32-bit AHB-Lite bus to access memory and memory-mapped peripherals (more detail in Section 2.1)
- The single-cycle IO block provides high-speed, deterministic access to GPIOs via each processor's IOPORT
- 26 system-level interrupts are routed to both processors
- A multi-drop Serial Wire Debug bus provides debug access to both processors from an external debug host

2.3.1. SIO

The Single-cycle IO block (SIO) contains several peripherals that require low-latency, deterministic access from the processors. It is accessed via each processor's IOPORT: this is an auxiliary bus port on the Cortex-M0+ which can perform rapid 32-bit reads and writes. The SIO has a dedicated bus interface for each processor's IOPORT, as shown in Figure 7. Processors access their IOPORT with normal load and store instructions, directed to the special IOPORT address segment, `0xd0000000···0xdfffffff`. The SIO appears as memory-mapped hardware within the IOPORT space.

NOTE

The SIO is not connected to the main system bus due to its tight timing requirements. It can only be accessed by the processors, or by the debugger via the processor debug ports.

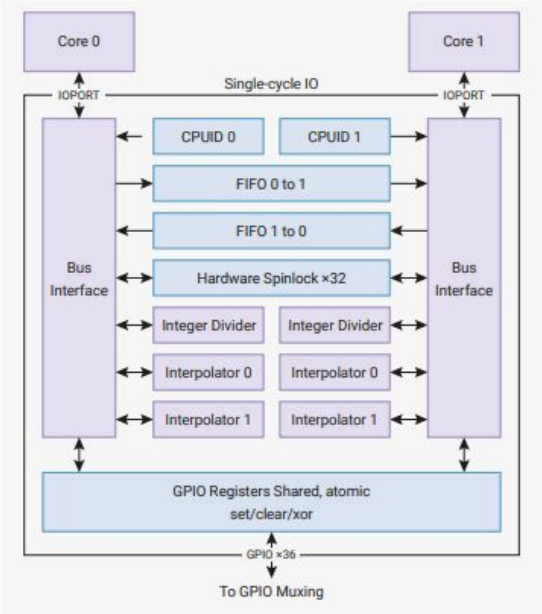


Figure 7. The single-cycle IO block contains memory-mapped hardware which the processors must be able to access quickly. The FIFOs and spinlocks support message passing and synchronisation between the two cores. The shared GPIO registers provide fast and concurrency-safe direct access to GPIO-capable pins. Some core-local arithmetic hardware can be used to accelerate common tasks on the processors.

All IOPORT reads and writes (and therefore all SIO accesses) take place in exactly one cycle, unlike the main AHB-Lite system bus, where the Cortex-M0+ requires two cycles for a load or store, and may have to wait longer due to contention from other system bus masters. This is vital for interfaces such as GPIO, which have tight timing requirements.

SIO registers are mapped to word-aligned addresses in the range `0xd0000000···0xd000017c`. The remainder of the IOPORT space is reserved for future use.

The SIO peripherals are described in more detail in the following sections.

2.3.1.1. CPUID

The register CPUID is the first register in the IOPORT space. Core 0 reads a value of 0 when accessing this address, and

2.3.1.2. GPIO Control

The processors have access to GPIO registers for fast and direct control of pins with GPIO functionality. There are two identical sets of registers:

- `GPIO_x` for direct control of IO bank 0 (user GPIOs 0 to 29, starting at the LSB)
- `GPIO_HI_x` for direct control of the QSPI IO bank (in the order SCLK, SSn, SD0, SD1, SD2, SD3, starting at the LSB)

NOTE

To drive a pin with the SIO's GPIO registers, the GPIO multiplexer for this pin must first be configured to select the SIO GPIO function. See Table 279.

These GPIO registers are *shared* between the two cores, and both cores can access them simultaneously. There are three registers for each bank:

- Output registers, GPIO_OUT and GPIO_HI_OUT, are used to set the output level of the GPIO (1/0 for high/low)
- Output enable registers, GPIO_OE and GPIO_HI_OE, are used to enable the output driver. 0 for high-impedance, 1 for drive high/low based on GPIO_OUT and GPIO_HI_OUT.
- Input registers, GPIO_IN and GPIO_HI_IN, allow the processor to sample the current state of the GPIOs

Reading GPIO_IN returns all 30 GPIO values (or 6 for GPIO_HI_IN) in a single read. Software can then mask out individual pins it is interested in.

SDK: https://github.com/raspberrypi/pico-sdk/blob/master/src/rp2_common/hardware_gpio/include/hardware/gpio.h Lines 674 - 676

```
674 static inline bool gpio_get(uint gpio) {
675     return !!((1ul << gpio) & sio_hw->gpio_in);
676 }
```

The `OUT` and `OE` registers also have atomic SET, CLR, and XOR aliases, which allows software to update a subset of the pins in one operation. This is vital not only for safe parallel GPIO access between the two cores, but also safe concurrent GPIO access in an interrupt handler and foreground code running on one core.

SDK: https://github.com/raspberrypi/pico-sdk/blob/master/src/rp2_common/hardware_gpio/include/hardware/gpio.h Lines 696 - 698

```
696 static inline void gpio_set_mask(uint32_t mask) {
697     sio_hw->gpio_set = mask;
698 }
```

SDK: https://github.com/raspberrypi/pico-sdk/blob/master/src/rp2_common/hardware_gpio/include/hardware/gpio.h Lines 705 - 707

```
705 static inline void gpio_clr_mask(uint32_t mask) {
706     sio_hw->gpio_clr = mask;
707 }
```

2.3.1.2. GPIO Control

The processors have access to GPIO registers for fast and direct control of pins with GPIO functionality. There are two identical sets of registers:

- `GPIO_x` for direct control of IO bank 0 (user GPIOs 0 to 29, starting at the LSB)
- `GPIO_HI_x` for direct control of the QSPI IO bank (in the order SCLK, SSn, SD0, SD1, SD2, SD3, starting at the LSB)

SDK: https://github.com/raspberrypi/pico-sdk/blob/master/src/rp2_common/hardware_gpio/include/hardware/gpio.h Lines 696 - 698

```
696 static inline void gpio_set_mask(uint32_t mask) {
697     sio_hw->gpio_set = mask;
698 }
```

SDK: https://github.com/raspberrypi/pico-sdk/blob/master/src/rp2_common/hardware_gpio/include/hardware/gpio.h Lines 705 - 707

```
705 static inline void gpio_clr_mask(uint32_t mask) {
706     sio_hw->gpio_clr = mask;
707 }
```

SDK: https://github.com/raspberrypi/pico-sdk/blob/master/src/rp2_common/hardware_gpio/include/hardware/gpio.h Lines 705 - 707

```
705 static inline void gpio_clr_mask(uint32_t mask) {
706     sio_hw->gpio_clr = mask;
707 }
```

SIO: GPIO_OUT Register

Offset: 0x010

Description

GPIO output value

Bits	Description	Type	Reset
29:0	Set output level (1/0 → high/low) for GPIO0...29. Reading back gives the last value written, NOT the input value from the pins. If core 0 and core 1 both write to GPIO_OUT simultaneously (or to a SET/CLR/XOR alias), the result is as though the write from core 0 took place first, and the write from core 1 was then applied to that intermediate result.	RW	0x00000000

SIO: GPIO_OUT Register

Offset: 0x010

Description

GPIO output value

2.3.1.7. List of Registers

The SIO registers start at a base address of `0xd0000000` (defined as SIO_BASE in SDK).

Bits	Description	Type	Reset
29:0	Set output level (1/0 → high/low) for GPIO0...29. Reading back gives the last value written, NOT the input value from the pins. If core 0 and core 1 both write to GPIO_OUT simultaneously (or to a SET/CLR/XOR alias), the result is as though the write from core 0 took place first, and the write from core 1 was then applied to that intermediate result.	RW	0x00000000

Turning a LED on with Go

```
package main

func main() {
   const ledPin = 25
   const addrSIO = 0xd0000000
   const offGPIO = 0x010
   var gpiooutreg *uint32 = addrSIO+offGPIO
   *gpioutreg = 1 << ledPin
}
```

Turning a LED on with Go - Memory Mapped IO and Volatility

```
import "unsafe"

func writeRegister(reg uintptr, value uint32) {
   ptr := (*uint32)(unsafe.Pointer(reg))
   *ptr = value
}
```

Turning a LED on with Go

```
package main

func main() {
   const ledPin = 25
   const addrSIO = 0xd0000000
   const offGPIO = 0x10
   // Set Output ON.
   writeRegister(addrSIO+offGPIO, 1<<ledPin)
}
```

SIO: GPIO_OE Register

Offset: 0x020

Description

GPIO output enable

Bits	Description	Type	Reset
29:0	Set output enable (1/0 → output/input) for GPIO0...29. Reading back gives the last value written. If core 0 and core 1 both write to GPIO_OE simultaneously (or to a SET/CLR/XOR alias), the result is as though the write from core 0 took place first, and the write from core 1 was then applied to that intermediate result.	RW	0x00000000

2.19.5.1. Select an IO function

An IO pin can perform many different functions and must be configured before use. For example, you may want it to be a `UART_TX` pin, or a `PWM` output. The SDK provides `gpio_set_function` for this purpose. Many SDK examples will call `gpio_set_function` at the beginning so that it can print to a UART.

2.19.6.1. IO - User Bank

The User Bank IO registers start at a base address of `0x40014000` (defined as IO_BANK0_BASE in SDK).

IO_BANK0: GPIO0_CTRL, GPIO1_CTRL, ..., GPIO28_CTRL, GPIO29_CTRL Registers

Offsets: 0x004, 0x00c, ..., 0x0e4, 0x0ec

Description

GPIO control including function select and overrides.

	Function								
GPIO	F1	F2	F3	F4	F5	F6	F7	F8	F9
25	SPI1 CSn	UART1 RX	I2C0 SCL	PWM4 B	SIO	PIO0	PIO1	CLOCK GPOUT3	USB VBUS DET

4:0	FUNCSEL	Function select. 31 == NULL. See GPIO function table for available functions.	RW	0x1f

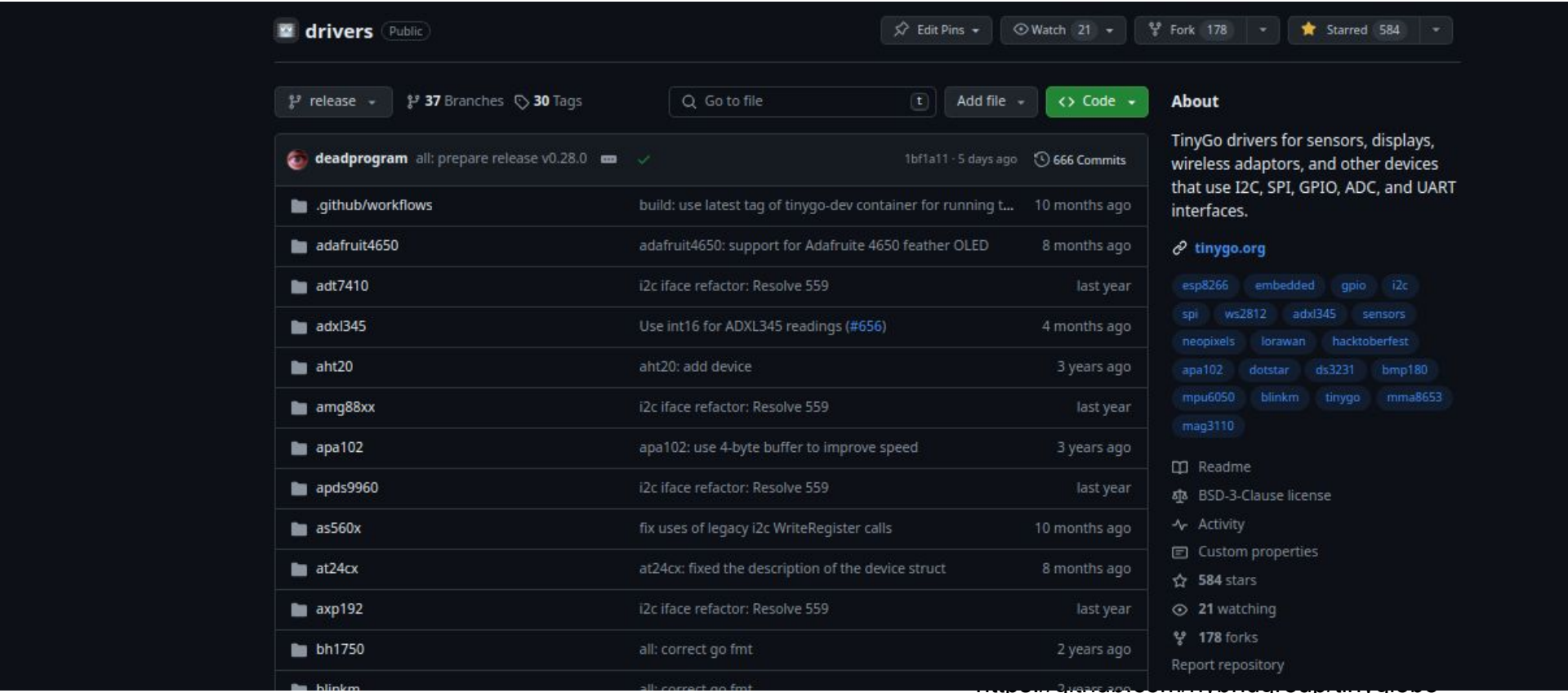
Turning a LED on with Go

```
func main() {
    const ledPin = 25
    const addrIObank = 0x40014000
    const addrSIO = 0xd0000000
    const offGPIO = 0x10
    const offEnable = 0x20
    const sizeIOPinBlk = 4 * 2
    const offIOPinCtrl = 4
    const fnIOGPIO = 5
    // Enable GPIO function in the pin Control IO bank.
    writeRegister(addrIObank+ledPin*sizeIOPinBlk+offIOPinCtrl, fnIOGPIO)
    // Enable digital output on that pin in the SIO.
    writeRegister(addrSIO+offEnable, 1<<ledPin)
    // Set Output ON.
    writeRegister(addrSIO+offGPIO, 1<<ledPin)
}
```

Intro to Embedded Systems

Mapped Memory

Mapped Memory



drivers Public

Edit Pins | Watch 21 | Fork 178 | Starred 584

release | 37 Branches | 30 Tags | Go to file t | Add file | Code

deadprogram all: prepare release v0.28.0 — 1bf1a11 · 5 days ago — 666 Commits

.github/workflows	build: use latest tag of tinygo-dev container for running t...	10 months ago
adafruit4650	adafruit4650: support for Adafruite 4650 feather OLED	8 months ago
adt7410	i2c iface refactor: Resolve 559	last year
adxl345	Use int16 for ADXL345 readings (#656)	4 months ago
aht20	aht20: add device	3 years ago
amg88xx	i2c iface refactor: Resolve 559	last year
apa102	apa102: use 4-byte buffer to improve speed	3 years ago
apds9960	i2c iface refactor: Resolve 559	last year
as560x	fix uses of legacy i2c WriteRegister calls	10 months ago
at24cx	at24cx: fixed the description of the device struct	8 months ago
axp192	i2c iface refactor: Resolve 559	last year
bh1750	all: correct go fmt	2 years ago
blinkm	all: correct go fmt	3 years ago

About

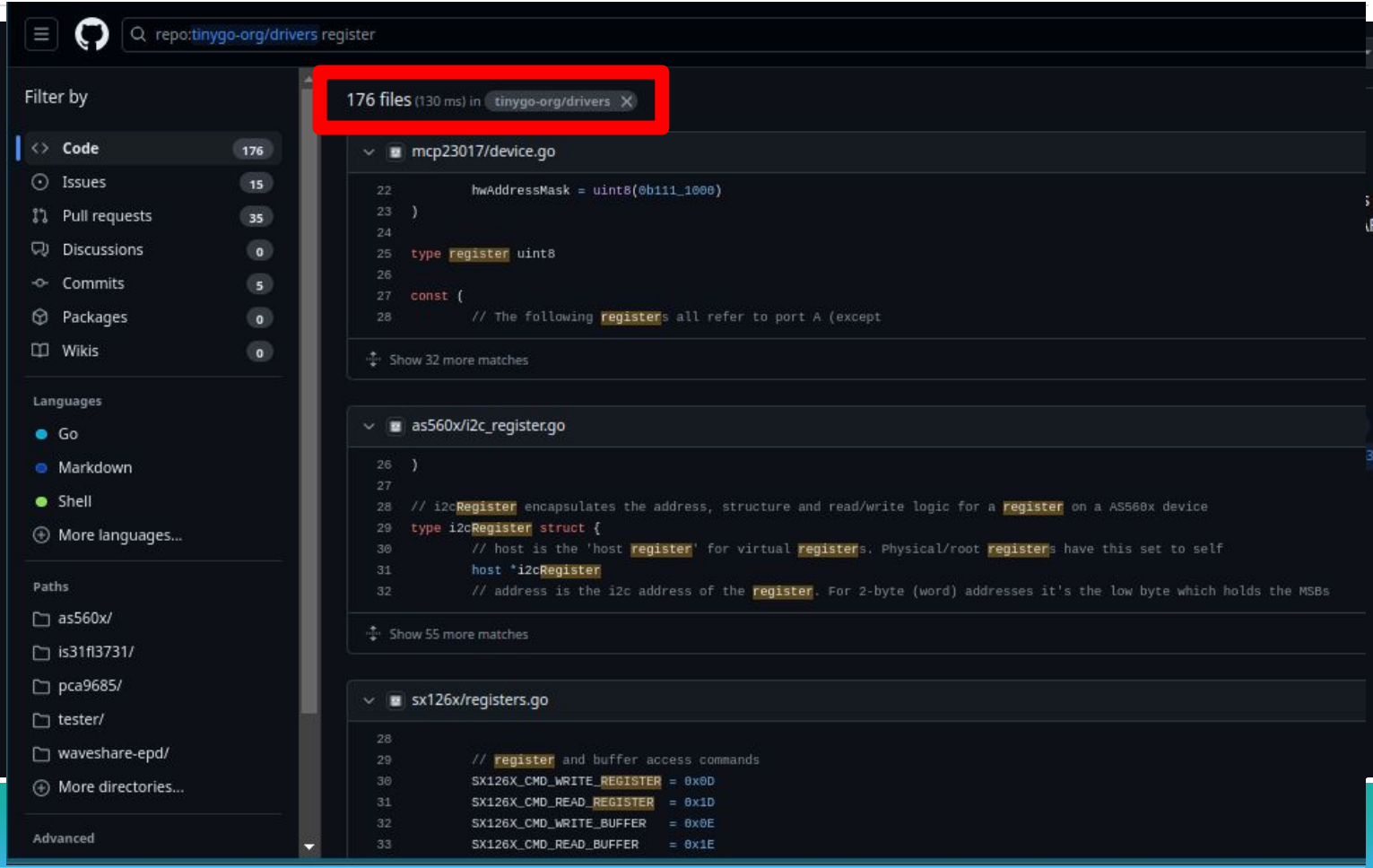
TinyGo drivers for sensors, displays, wireless adaptors, and other devices that use I2C, SPI, GPIO, ADC, and UART interfaces.

tinygo.org

esp8266 embedded gpio i2c spi ws2812 adxl345 sensors neopixels lorawan hacktoberfest apa102 dotstar ds3231 bmp180 mpu6050 blinkm tinygo mma8653 mag3110

Readme
BSD-3-Clause license
Activity
Custom properties
584 stars
21 watching
178 forks
Report repository

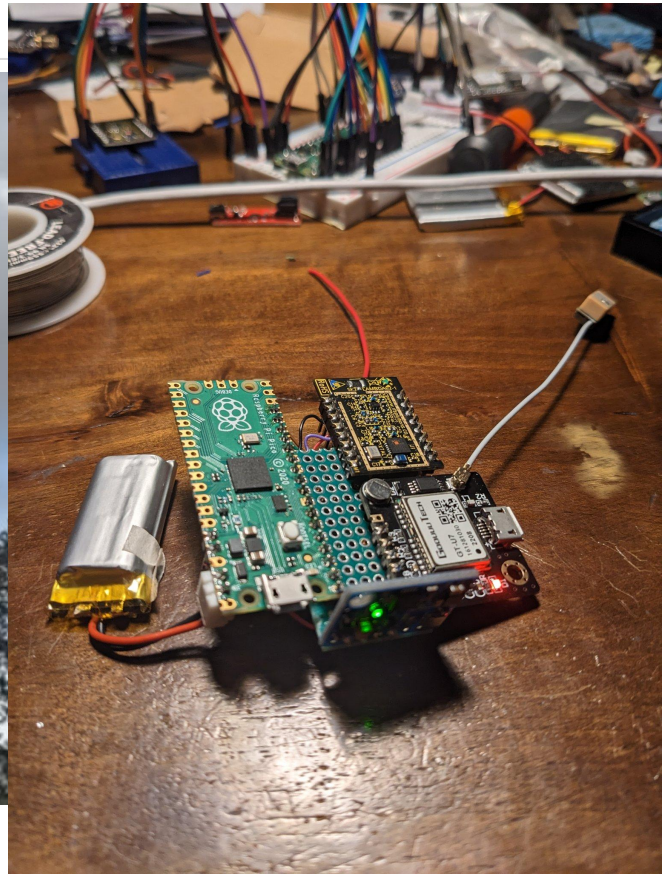
Mapped Memory



Intro to Embedded Systems

MCU applications

MCU Applications - TinyGlobo

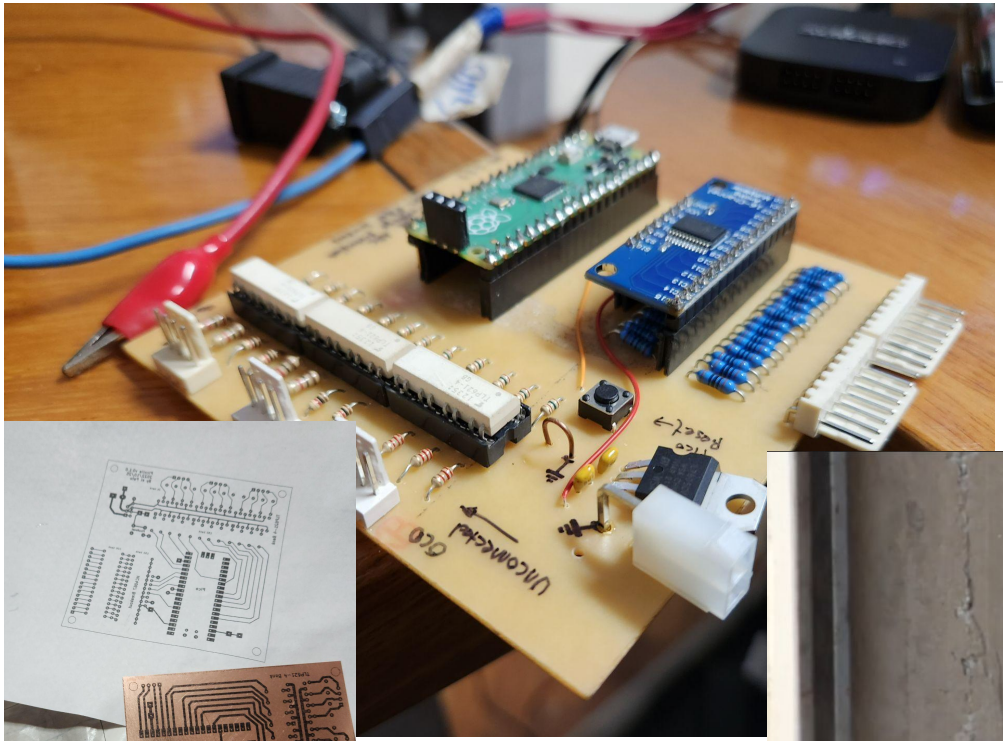


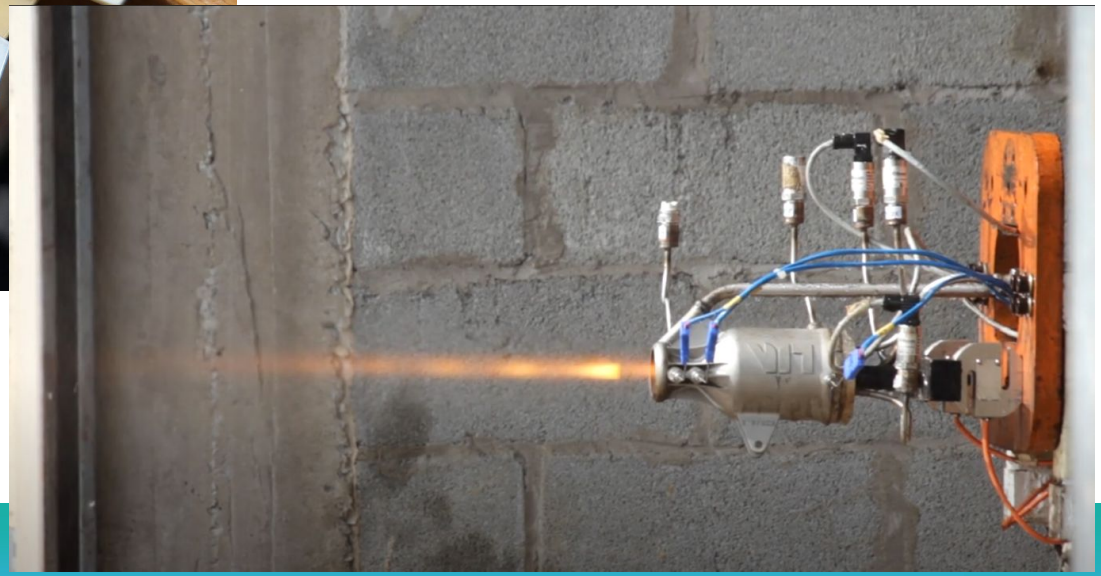
https://github.com/hybridgroup/tinyglobo

MCU Applications

LIA Aerospace

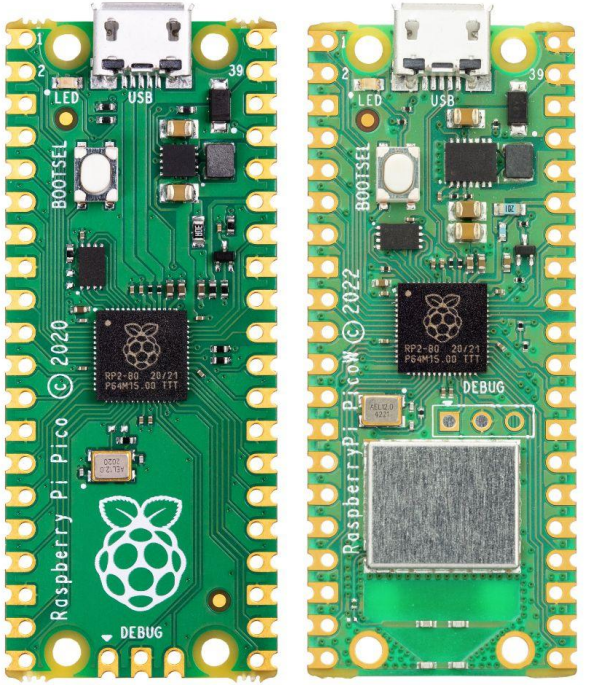
"**Picolia**" Instrumentation Board





MCU Applications - Pico W

- Wifi Driver:
  - github.com/soypat/cyw43439
- Ethernet/IP/TCP/HTTP stack:
  - github.com/soypat/seqs



Why?


LED
USB
BOOTSEL
DEBUG
RaspberryPi Pico W © 2022


Raspberry Pi 5
HDMI

02

Closing thoughts

Fin.

Thank you!

Information

Patricio Whittingslow
Github: **soypat**
Email: **graded.sp@gmail.com**
Twitter: @whittileaks

Figure 4. RP2040 bus fabric overview.