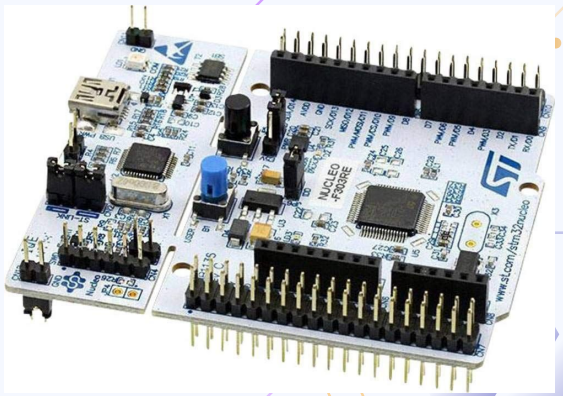
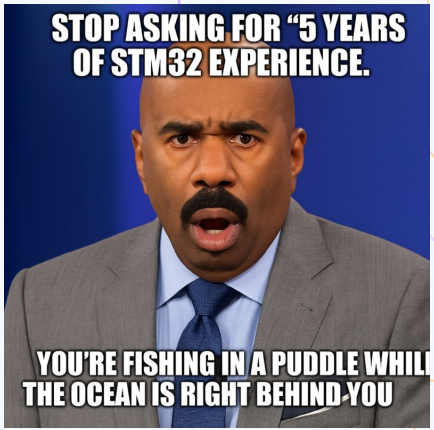
Intro to STM32: A HANDS-ON WORKSHOP

Attendance Form

IEEE Penn State - Projects Committee
SP26



WTF is an STM32?



- ARM Cortex-M chip
- Microcontroller
  - Big brain, bigger muscle
  - Made for **specific** tasks
- **Used in the industry**
  - *Tesla, Qualcomm, various corporations*
- WHY STM32?
  - Great to learn
  - Computational prowess
  - Versatility
  - Marketable skills…**WE** are getting jobs with this one 🤑

ANY embedded job that requires low-level C/C++ jobs -> STM32 skills transfer

This Flipper-Zero has an STM32 inside it

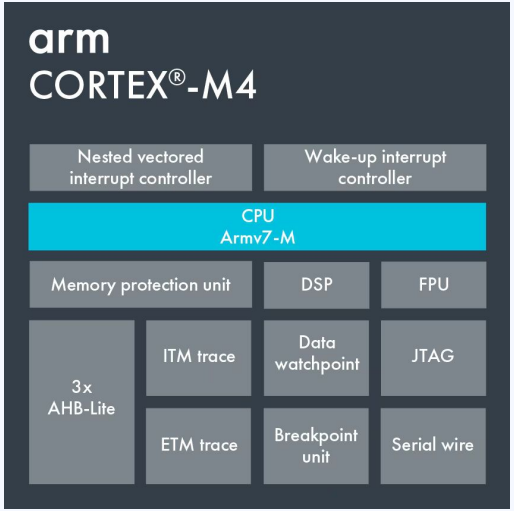
So does this one-wheeler too 😱

Part 1

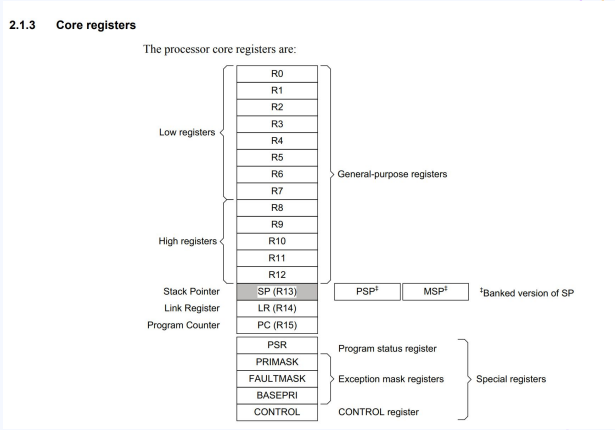
Essentials of Microcontrollers

Central Processing Unit (CPU)

- “Brain” of system
- ARM Cortex-M
  - “M” for microcontroller
- What it does:
  - *Executes instructions one-by-one*
  - *Manages flow of data*
  - *Math (computers run on math)*
    - Arithmetic Logical Unit (ALU)
  - Manages memory
    - ***Registers***
      - (This is a TYPE of memory)



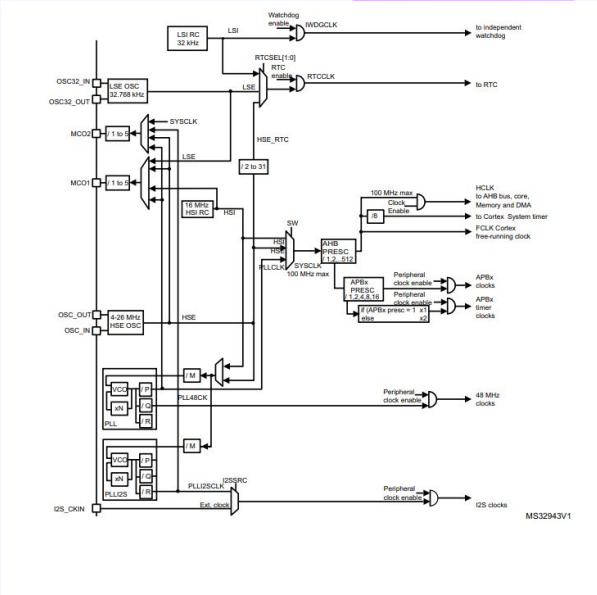
Registers



- Speedy memory with a certain width
  - Arduino -> 8-bit
  - STM32 -> **32-bit**
    - [31 . . . . . 0]
- Registers are *memory mapped* (MMIO)
- General purpose & peripheral registers
- **Hold critical information**
  - Variables, function returns, etc.
  - Used extensively for arithmetic operations
- Certain registers hold certain values
  - GPIOx_IDR -> GPIO register for input data
  - TIMx_CNT -> register for keeping count in a timer

Clocks

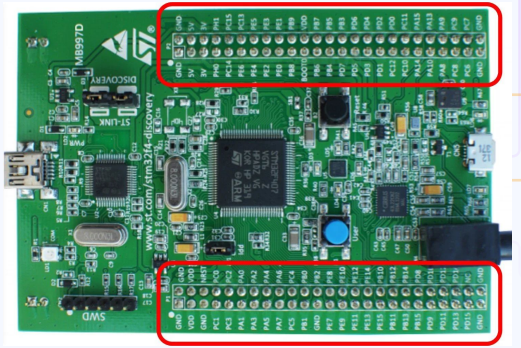
- **Drives the CPU**
  - State changes
  - Instructions executed every cycle
- Keeps tasks synchronous and in order
- Speed
  - F411RE: 84 MHz
  - Normal CPU: 3 to 5 GHZ
- RCC registers -> control system clocks
- Clocks for various GPIO groups
- Useful for:
  - Timers
  - ***Basically everything***

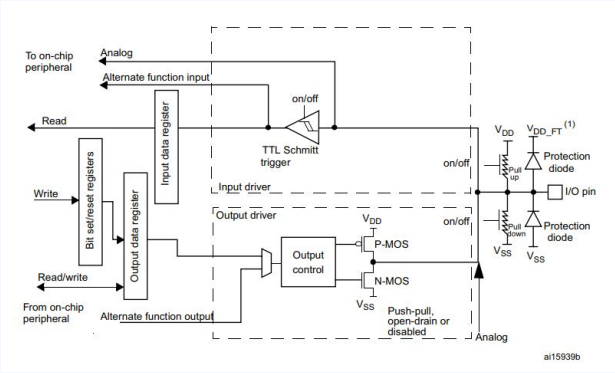


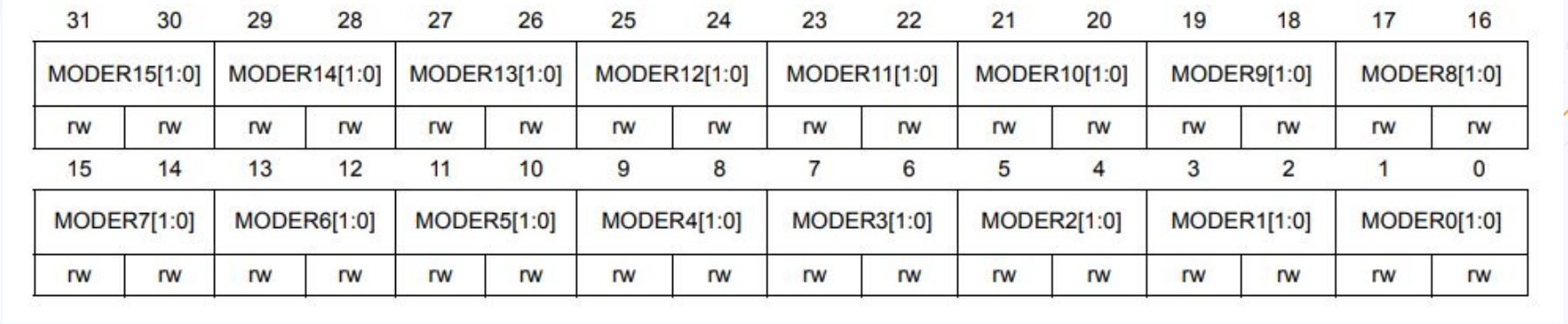
Don't worry, you don't need to know how this works

Input and Output (I/O)

- **General Purpose Input/Output pins (GPIO)**
- Main interface with outside world
- Logically grouped
  - *GPIOA, GPIOB, GPIOC*
- **Different modes**
  - *Input*
    - Pull-up & pull-down
  - *Output*
    - Push-pull & open-drain
  - *Alternate function (AF)*
  - *Analog*
    - Raw data (ADC)







31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

GPIO “MODER” register

- 2-bits
- Data
  - 00 - input mode
  - 01 - output mode
  - 10 - alternate function mode
  - 11 - analog mode

Programming the STM32

- Coding techniques (pick one!)
  - ***Bare metal***: accessing registers directly
    - Advanced; bit-level math
  - ***Low level***: thin abstraction around bare metal
    - No bit-level math needed
  - ***Hardware Abstraction Layer (HAL)***: high level interface
    - Easier to program with
    - Clean & portable
    - Slower (insignificant for our purposes)
- Programmed through ST-LINK

```
87    while ((FLASH->SR & FLASH_SR_BSY) != 0);
88    if ((FLASH->CR & FLASH_CR_LOCK) != 0)
89    {
90        FLASH->KEYR = 0x45670123;
91        FLASH->KEYR = 0xCDEF89AB;
92    }
93
94    FLASH->CR |= FLASH_CR_PG;
95    uint16_t toWrite = 0x5050;
96     *(uint32_t*)0x0801F000 = toWrite;
97     while ((FLASH->SR & FLASH_SR_BSY) != 0);
98     if ((FLASH->SR & FLASH_SR_EOP) != 0)
99     {
100     FLASH->SR = FLASH_SR_EOP;
101    }
102    FLASH->CR &= ~FLASH_CR_PG;
```

Programming flash memory in ***bare metal***

Initialization and de-initialization functions

This section contains the following APIs:

- *HAL_GPIO_Init()*
- *HAL_GPIO_DeInit()*

IO operation functions

This section contains the following APIs:

- *HAL_GPIO_ReadPin()*
- *HAL_GPIO_WritePin()*
- *HAL_GPIO_TogglePin()*
- *HAL_GPIO_LockPin()*
- *HAL_GPIO_EXTI_IRQHandler()*

HAL functions to interact with GPIO pins

Part 2

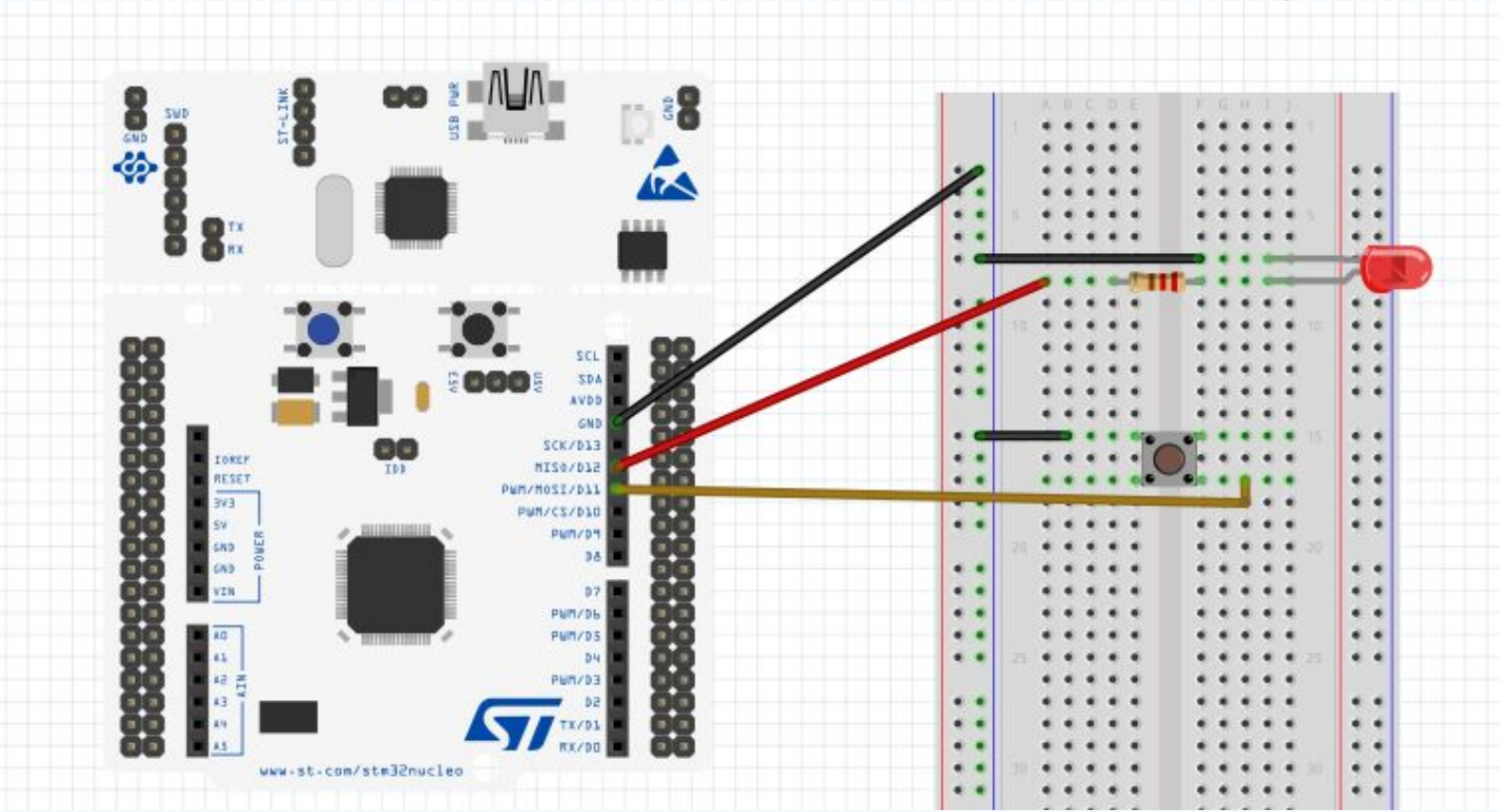
STM32CubeIDE and STM32CubeMX

IEEE GitHub Link:

https://github.com/psuieee/stm32-workshop1

IMPORTANT:

- **Make an ST account**
- Download **version 1.19.0**
  - **Click “select version”**


SWD
GND
ST-LINK
USB PWR
GND
TX
RX
SCL
SDA
AVDD
GND
SCK/D13
MISO/D12
PWM/MOSI/D11
PWM/CS/D10
PWM/D9
D8
D7
PWM/D6
PWM/D5
D4
PWM/D3
D2
TX/D1
RX/D0
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
www.st.com/stm32nucleo