MICRO-CONTROLLERS PART IV

BY JOGI

MICROCONTROLLER TALKS

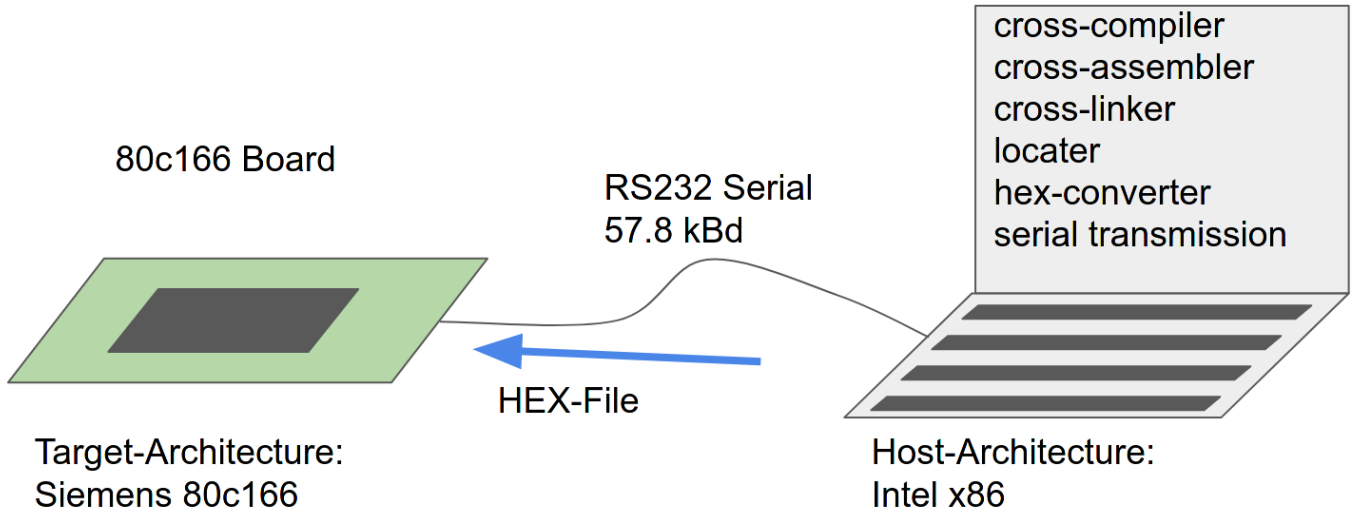
1. Introduction, history incl 6502, 80C166, via Arduino to Raspi Pico, Jogi, Code and Slides
2. MC-based learning-platforms, Micro:Bit & Calliope, Jogi, Code and Slides
3. ESP8266- and EPS32-Universe, Felix, Lightning Talks-Repo
4. **This** : More Pi Pico, Environments, Comparison, Eco-System, Jogi, Code and Slides
5. (unplanned) internal usecase RLX-Testfarm, making use of RubberJogi OSS, Lightning Talks Repo

- Recap first micro-controller talk
- Show the different envs for Pico
- Compare onboarding
- compare results
- Some infos ECO-System
- New : WLAN and BLE

SHORT RECAP

- Complicated setup with a lot of tools
- Arduino enters the world => all in one
- Raspberry Pi Pico => Python

PROGRAMMING ENVIRONMENT



STEPS NEEDED

```
1
2 Edit Code
3
4 Call Cross-C-Compiler (C ==> Assembler)
5
6 Call Cross-Assembler  (Assembler ==> Binary)
7
8 Call Cross-Linker (several Modules ==> one Executable)
9
10 Call Cross-Locator (knows Memory-Layout)
11
12 Call HEX-converter (Flash-Format)
13
14 Send via Serial line to Target-Board
15
```

STEPS NEEDED

```
1
2  Edit Code
3
4  Call Cross-C-Compiler (C ==> Assembler)
5
6  Call Cross-Assembler  (Assembler ==> Binary)
7
8  Call Cross-Linker (several Modules ==> one Executable)
9
10 Call Cross-Locator (knows Memory-Layout)
11
12 Call HEX-converter (Flash-Format)
13
14 Send via Serial line to Target-Board
15
```

STEPS NEEDED

```
1
2  Edit Code
3
4  Call Cross-C-Compiler (C ==> Assembler)
5
6  Call Cross-Assembler  (Assembler ==> Binary)
7
8  Call Cross-Linker (several Modules ==> one Executable)
9
10 Call Cross-Locator (knows Memory-Layout)
11
12 Call HEX-converter (Flash-Format)
13
14 Send via Serial line to Target-Board
15
```

STEPS NEEDED

```
 1
 2 Edit Code
 3
 4 Call Cross-C-Compiler (C ==> Assembler)
 5
 6 Call Cross-Assembler  (Assembler ==> Binary)
 7
 8 Call Cross-Linker (several Modules ==> one Executable)
 9
10 Call Cross-Locator (knows Memory-Layout)
11
12 Call HEX-converter (Flash-Format)
13
14 Send via Serial line to Target-Board
15
```

STEPS NEEDED

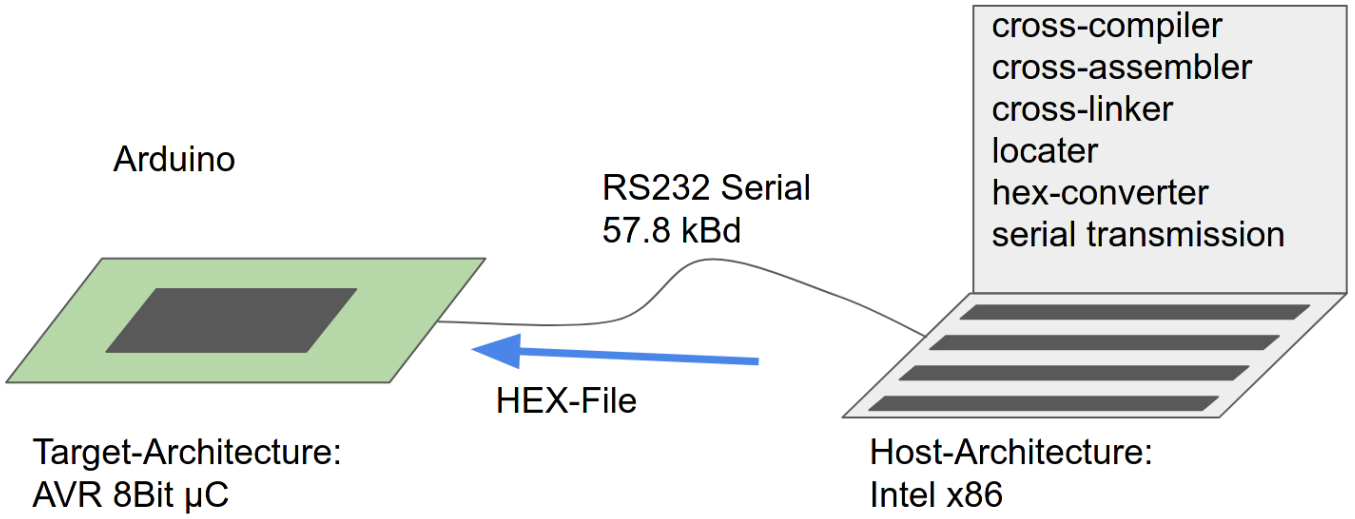
```
1
2  Edit Code
3
4  Call Cross-C-Compiler (C ==> Assembler)
5
6  Call Cross-Assembler  (Assembler ==> Binary)
7
8  Call Cross-Linker (several Modules ==> one Executable)
9
10 Call Cross-Locator (knows Memory-Layout)
11
12 Call HEX-converter (Flash-Format)
13
14 Send via Serial line to Target-Board
15
```

STEPS NEEDED

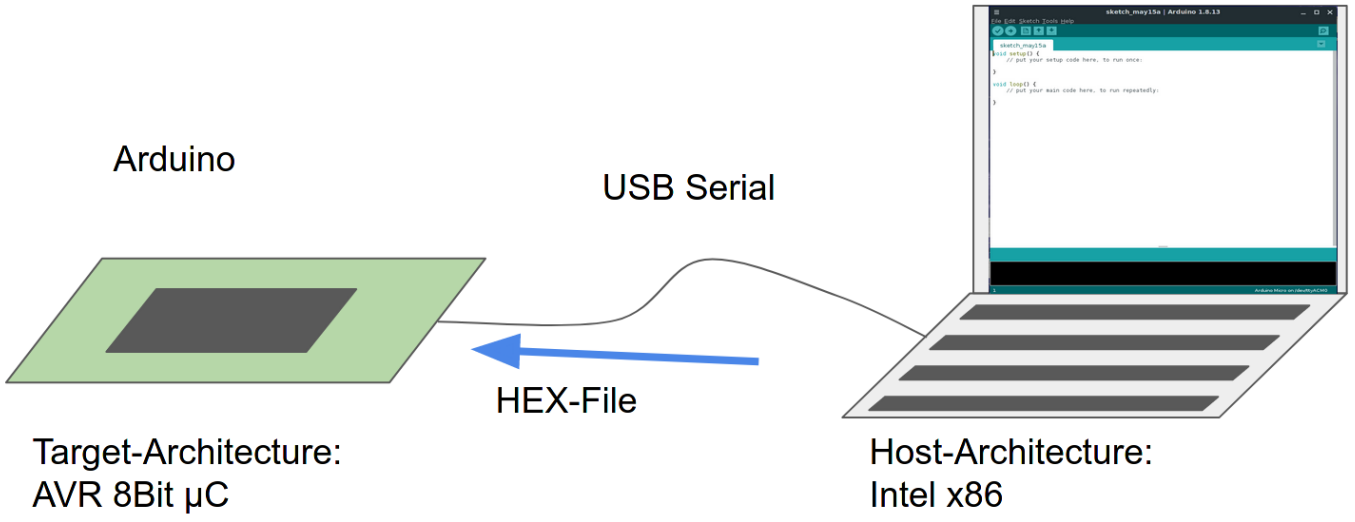
```
1
2  Edit Code
3
4  Call Cross-C-Compiler (C ==> Assembler)
5
6  Call Cross-Assembler  (Assembler ==> Binary)
7
8  Call Cross-Linker (several Modules ==> one Executable)
9
10 Call Cross-Locator (knows Memory-Layout)
11
12 Call HEX-converter (Flash-Format)
13
14 Send via Serial line to Target-Board
15
```

2005/2006 ARDUINO ENTERS THE WORLD

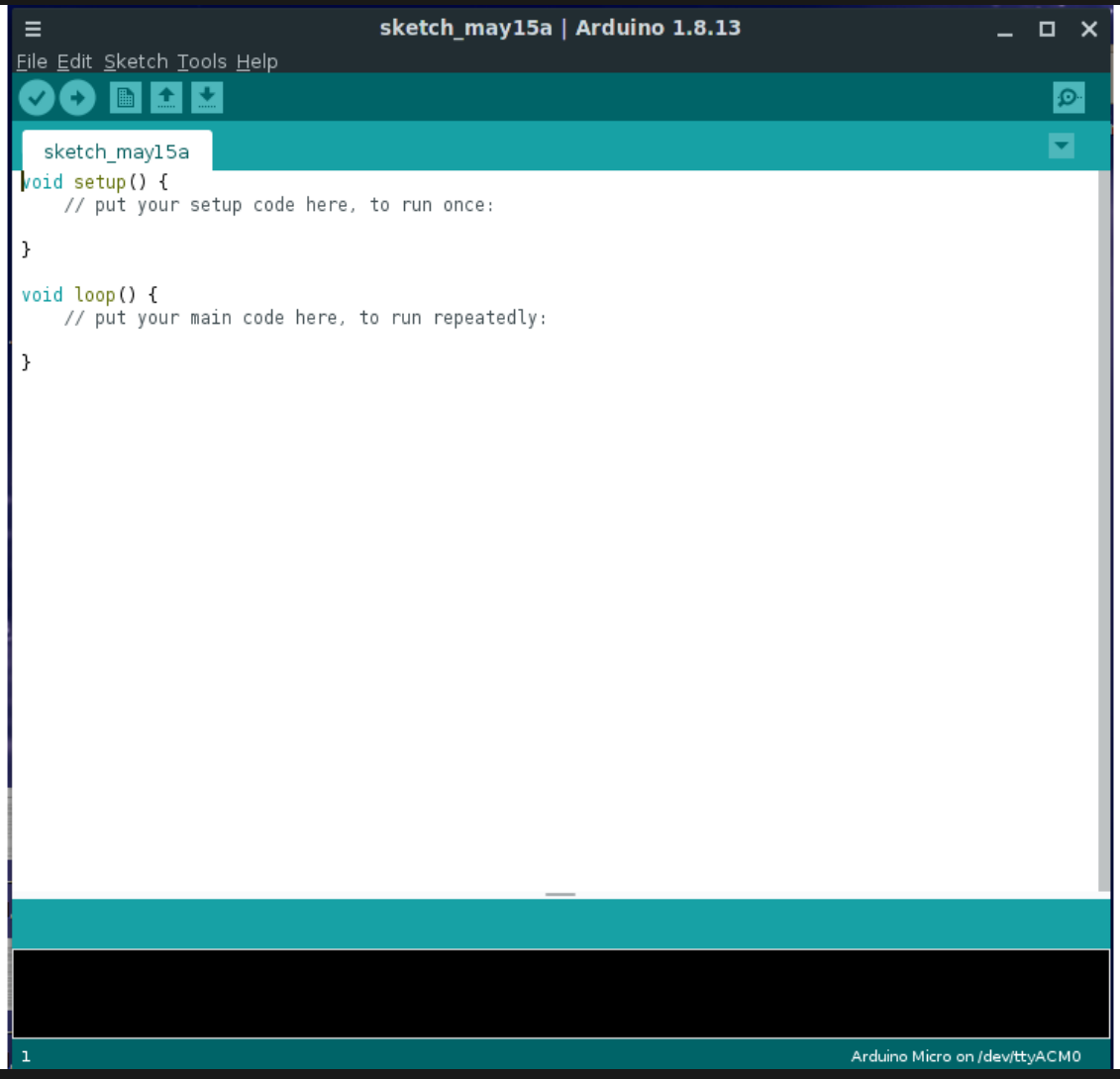
"THE" INNOVATION : FROM



TO "ARDUINO: ALL IN ONE"



THE IDE


sketch_may15a | Arduino 1.8.13
File Edit Sketch Tools Help
sketch_may15a
void setup() {
// put your setup code here, to run once:
}
void loop() {
// put your main code here, to run repeatedly:
}
1
Arduino Micro on /dev/ttyACM0

ARDUINO-COMPATIBILITY

- From 2005 till 2021
- Micro-Controller for "everyone"
- "needed" to be Arduino-compatible

2021 RASPBERRY PI PICO ENTERS THE WORLD

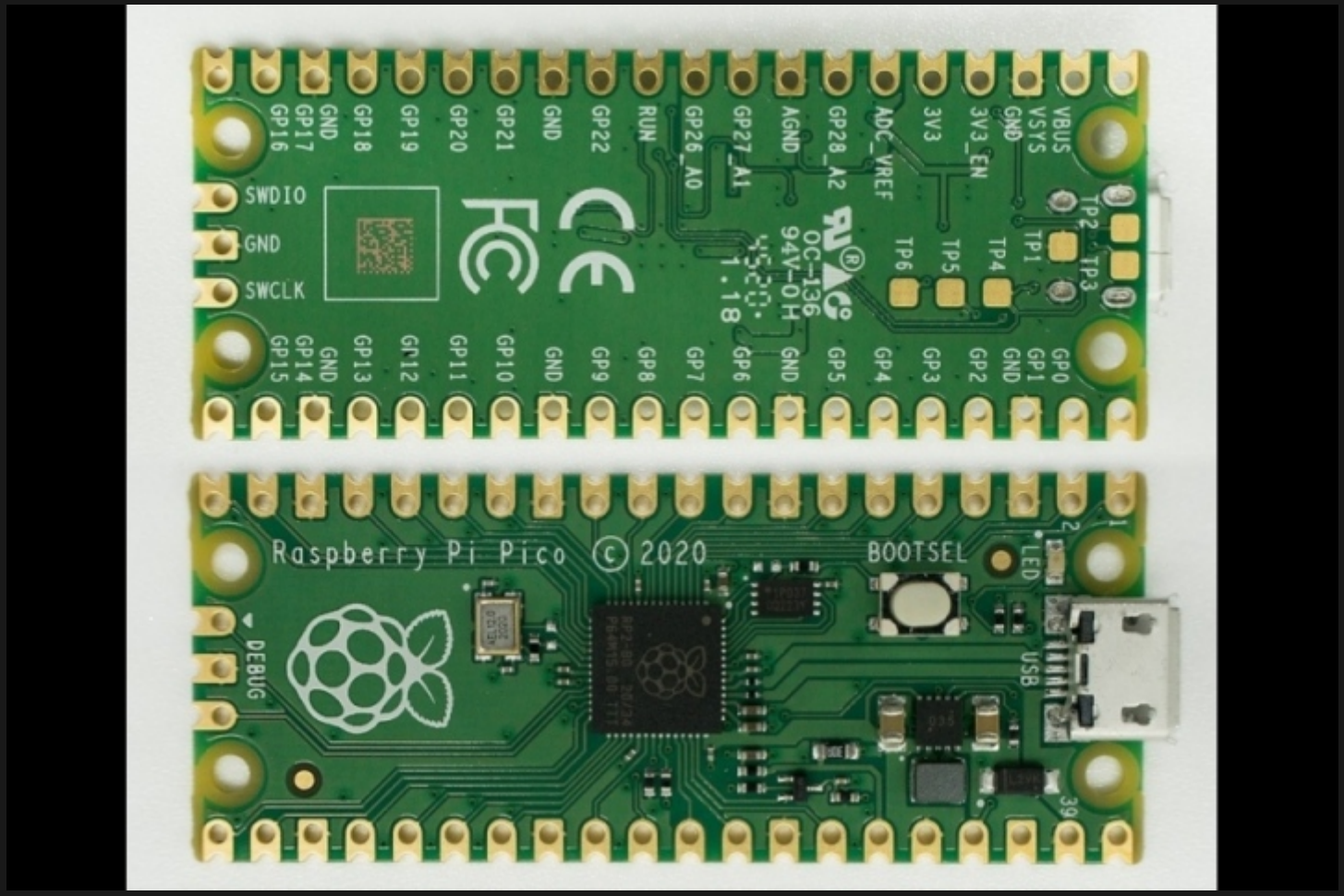
RASPBERRY PI PICO

- just another Micro-Controller
- But 3 interesting aspects
  - is from the Raspberry-PI foundation => might attract new "customers"
  - breaks with the "Has to run with Arduino-IDE"-Dogma
  - Has at least one interesting HW-Block, the PIO

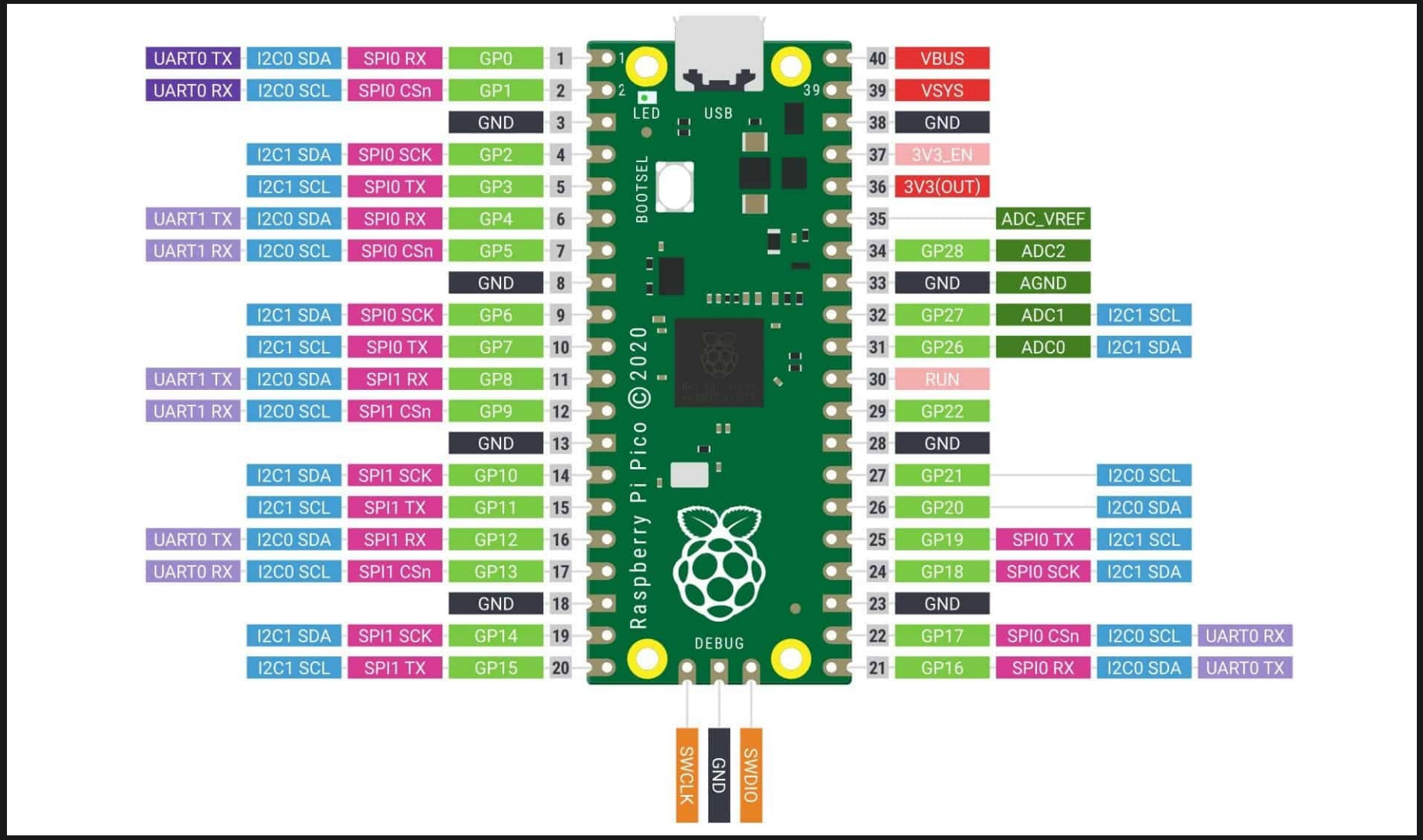
RASPBERRY PI PICO

- always starts as UF2-Board
- looks like a USB-Stick
- Microsoft-defined USB-Format
- especially dedicated for downloading firmware to MC via USB
- Cannot put just e.g. python-Files on it

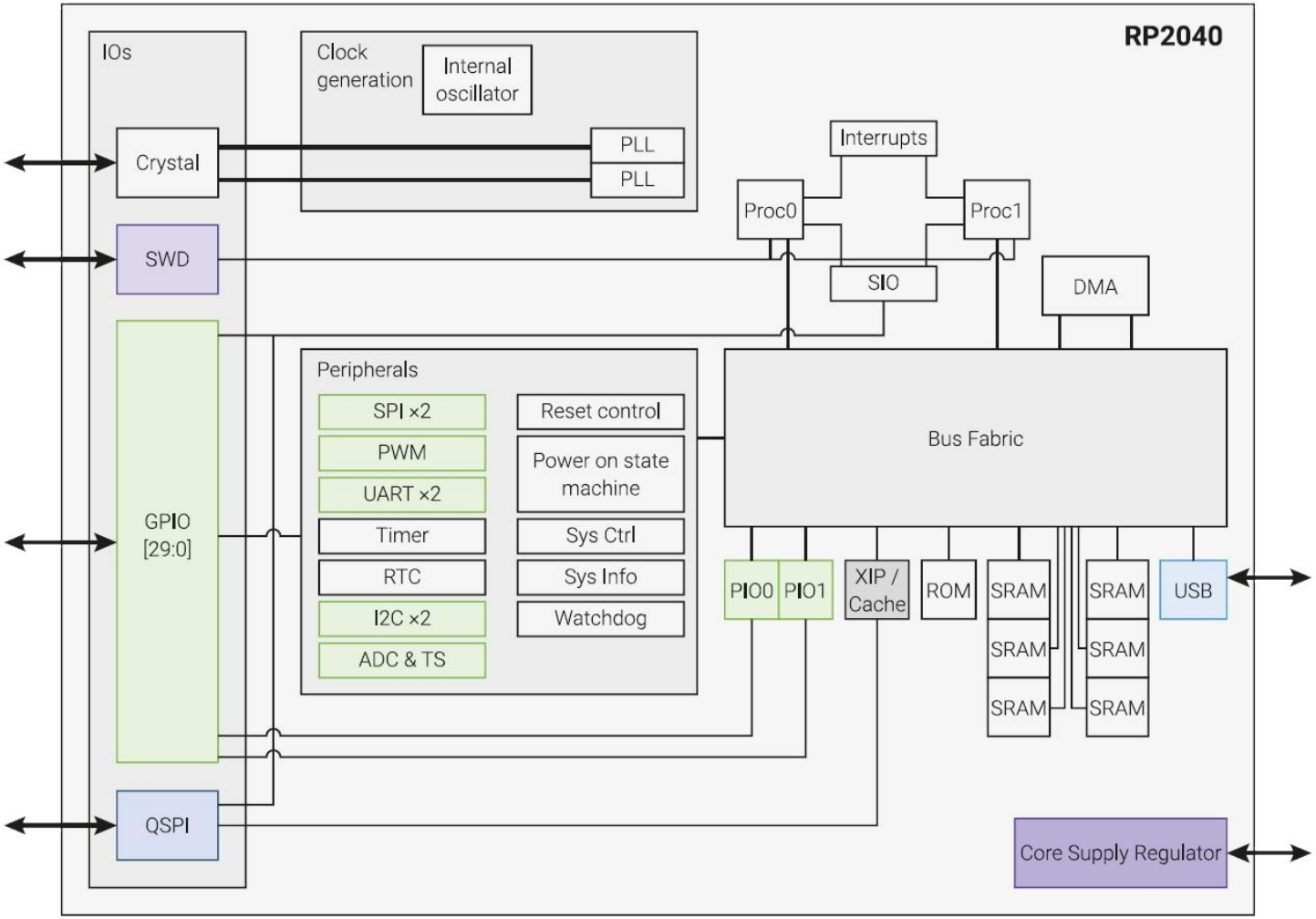
THE PICO ITSELF



THE PINS OF PICO



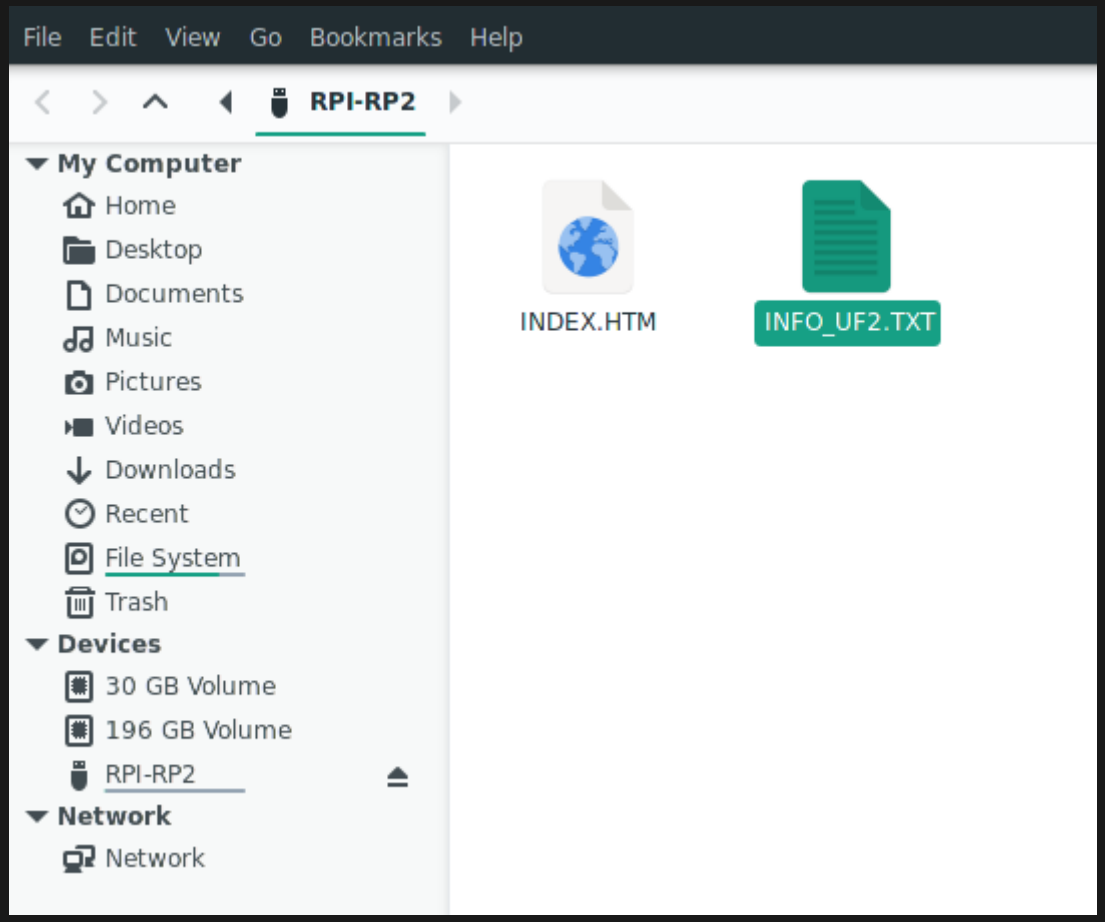
THE BLOCKS OF PICO



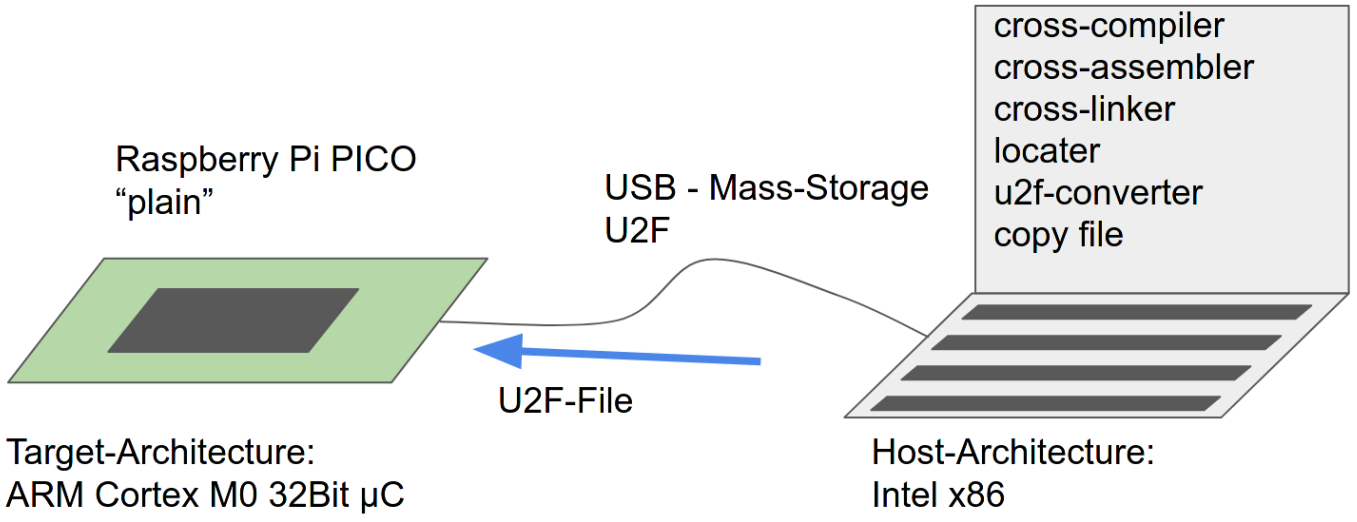
RASPI PICO DEV ENVIRONEMENTS

- Plain SDK (Assembler/C/C++)
- Micropython
- Circuitpython
- Arduino-IDE
- MMBasic

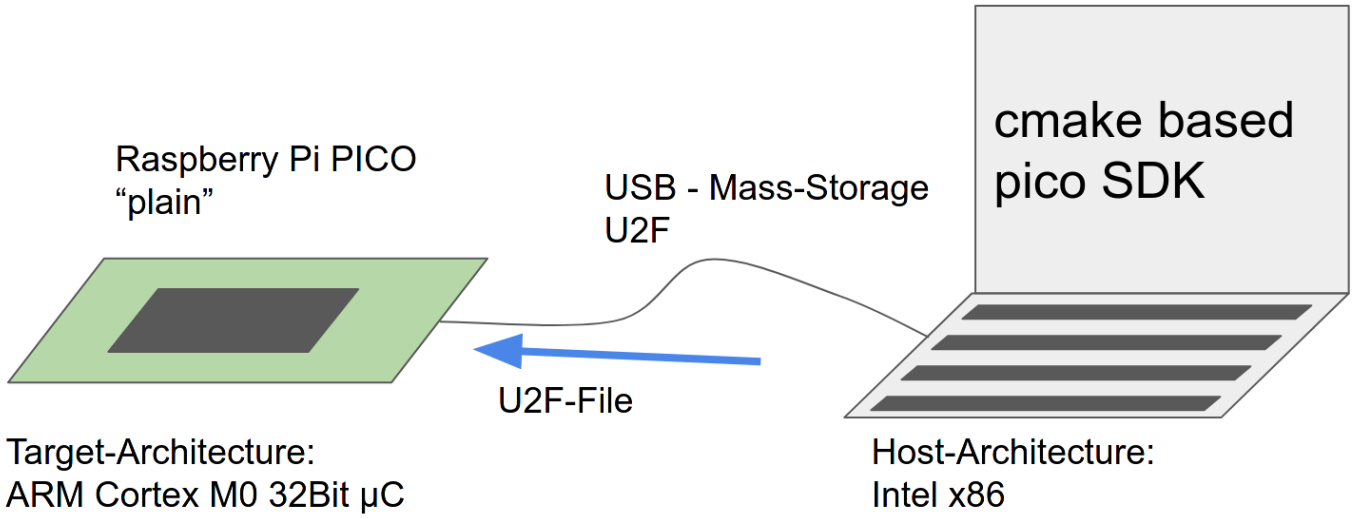
PICO AS U2F-USB MASS-STORAGE



DEV-ENVIRONMENT : SDK



DEV-ENVIRONEMENT : SDK (CMAKE)



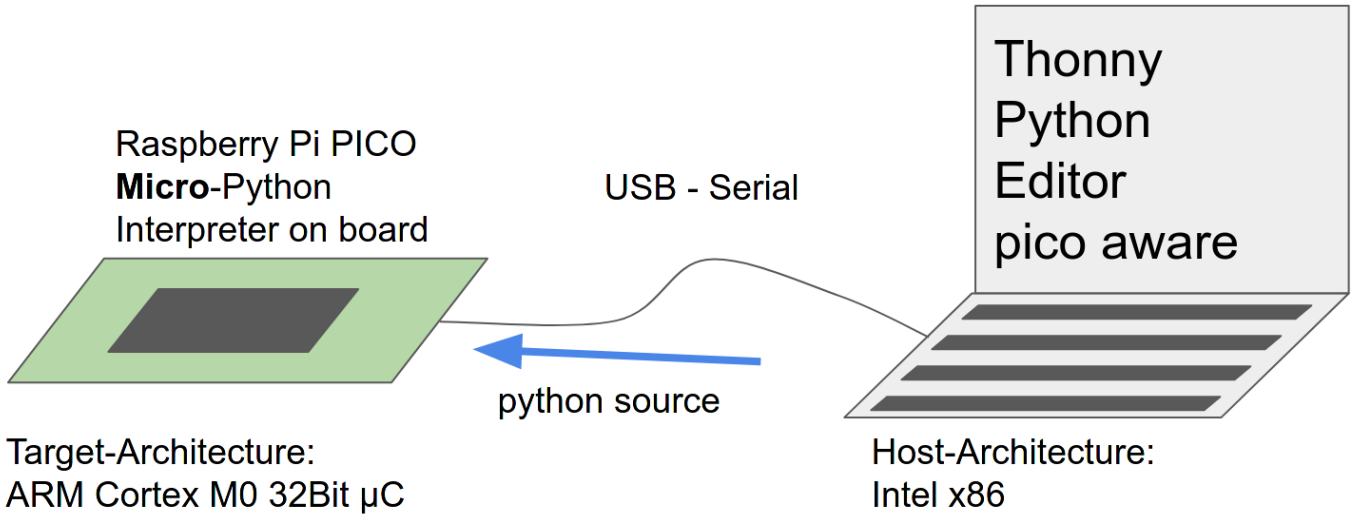
PYTHON

- Micro-Python
- Circuit Python

MICRO-PYTHON

- Attempt to put Python on Micro-Controller-Boards
- Started with an own Board
- Interpreter now available for a lot of Micro-C-Boards
- Including the Raspberry-Pi Pico

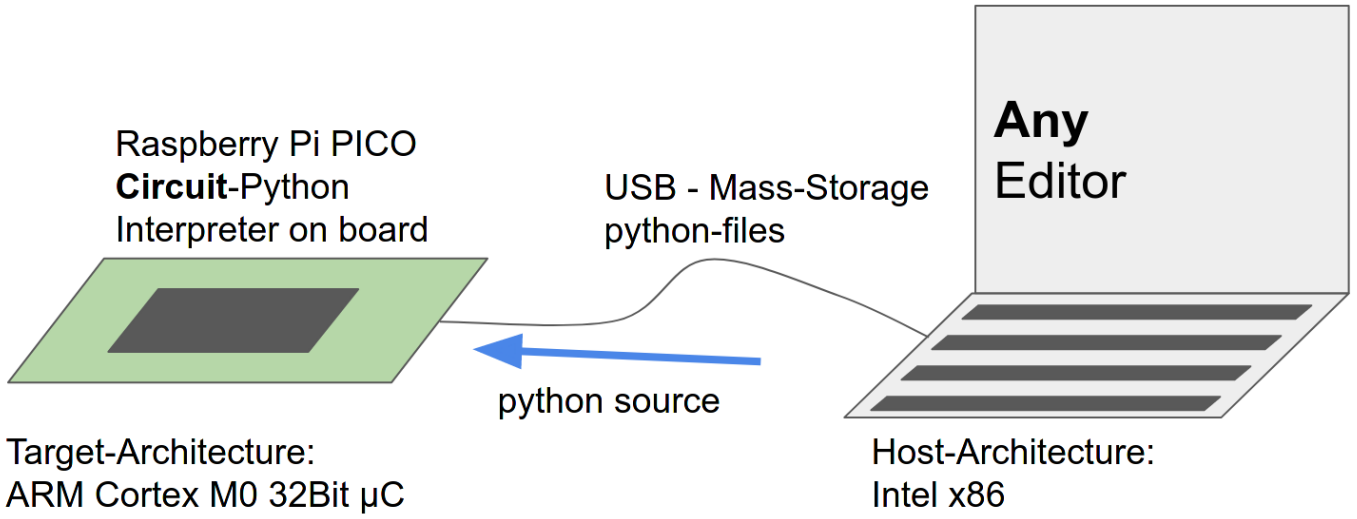
DEV-ENV MICRO-PYTHON



CIRCUIT-PYTHON

- Is a fork of Micro-Python
- Done/maintained by Adafruit
- A lot of differences in the libraries
- Main difference visible : Provides a "real" *USB-Drive*
- Can be used with *any* editor
- mu-editor is preffered, offers some capabilities

DEV-ENV CIRCUIT-PYTHON



MICRO-PYTHON : OFFICIAL

SETUP PART 1

- Download Micropython U2F (with or w/o WLAN)
- press BOOTSEL - Button
- connect USB
- A USB-Stick appears
- copy U2F to USB-Stick
- done

MICRO-PYTHON : OFFICIAL

SETUP PART 2

- Download Thonny
- Install Thonny
- (contains already serial tty-connection to Pico)
- done

MICRO-PYTHON : OFFICIAL

START WORKING

- Thonny == your IDE
- Connect to Pico
- Open file local or from pico
- Run

SETUP FROM RASPI-FOUNDATION

MICRO PYTHON

INTERNAL LED (PIN25)

```
1
2 import machine
3
4 led_onboard = machine.Pin(25, machine.Pin.OUT)
5
6 while True:
7     led_onboard.value(1)
8     led_onboard.value(0)
9
```

MICRO PYTHON

INTERNAL LED (PIN25)

```
1
2 import machine
3
4 led_onboard = machine.Pin(25, machine.Pin.OUT)
5
6 while True:
7     led_onboard.value(1)
8     led_onboard.value(0)
9
```

MICRO PYTHON

INTERNAL LED (PIN25)

```
1
2 import machine
3
4 led_onboard = machine.Pin(25, machine.Pin.OUT)
5
6 while True:
7     led_onboard.value(1)
8     led_onboard.value(0)
9
```

CIRCUIT PYTHON

INTERNAL LED (PIN25)

```
 1
 2 import board
 3 import digitalio
 4 import time
 5
 6 led = digitalio.DigitalInOut(board.LED)
 7 led.direction = digitalio.Direction.OUTPUT
 8
 9 while True:
10     led.value = True
11     time.sleep(0.5)
12     led.value = False
13     time.sleep(0.5)
14
15
```

CIRCUIT PYTHON

INTERNAL LED (PIN25)

```
 1
 2 import board
 3 import digitalio
 4 import time
 5
 6 led = digitalio.DigitalInOut(board.LED)
 7 led.direction = digitalio.Direction.OUTPUT
 8
 9 while True:
10     led.value = True
11     time.sleep(0.5)
12     led.value = False
13     time.sleep(0.5)
14
15
```

CIRCUIT PYTHON

INTERNAL LED (PIN25)

```
 1
 2 import board
 3 import digitalio
 4 import time
 5
 6 led = digitalio.DigitalInOut(board.LED)
 7 led.direction = digitalio.Direction.OUTPUT
 8
 9 while True:
10     led.value = True
11     time.sleep(0.5)
12     led.value = False
13     time.sleep(0.5)
14
15
```

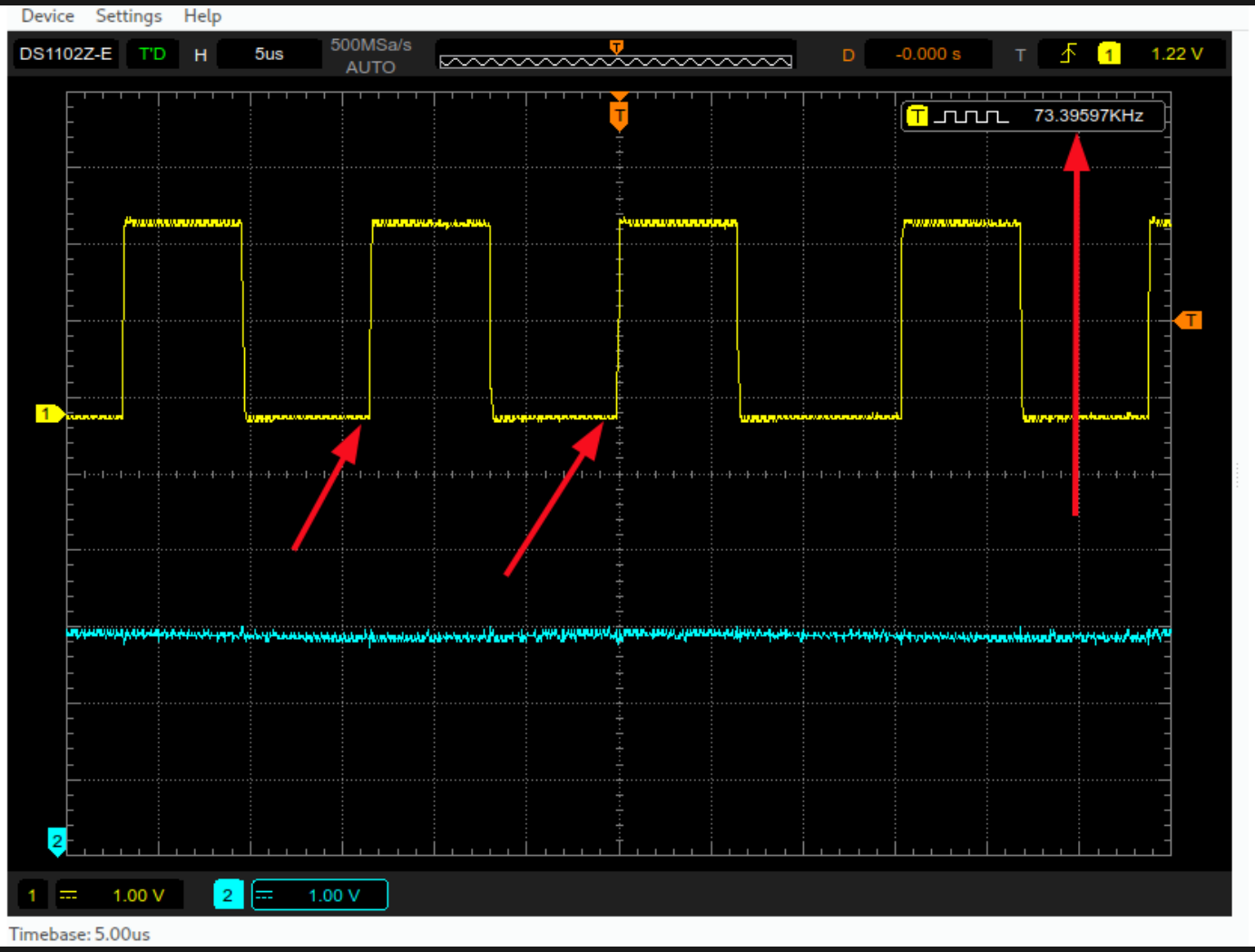
DIFFERENCES

- Libs completely differ
- Support for Hardware
- USB-appearance
- Autostart:
  - main.py Micropython
  - code.py Circuitpython
- Micropython is "official"

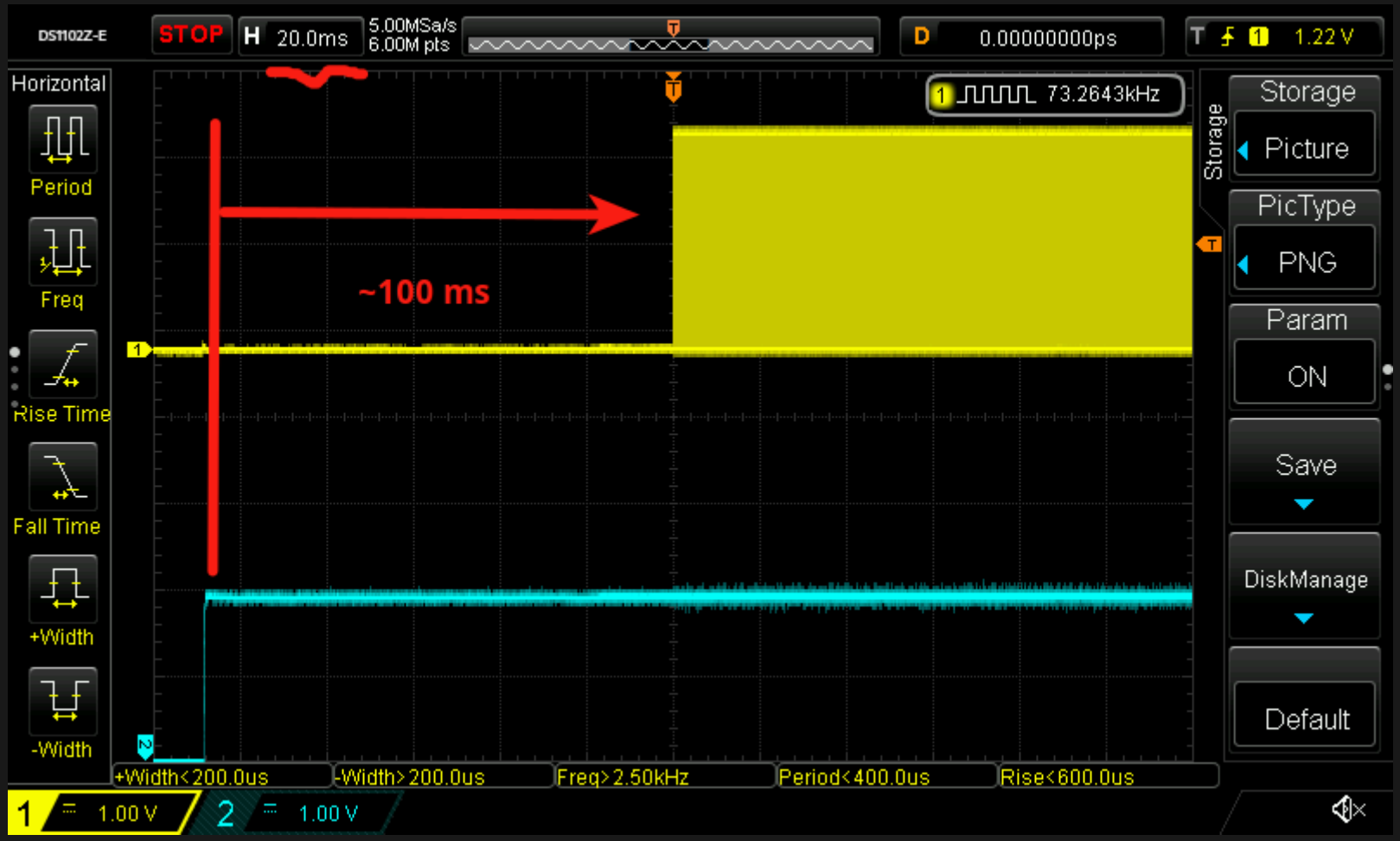
DEMO-TIME

WHAT WE (HOPEFULLY) SAW

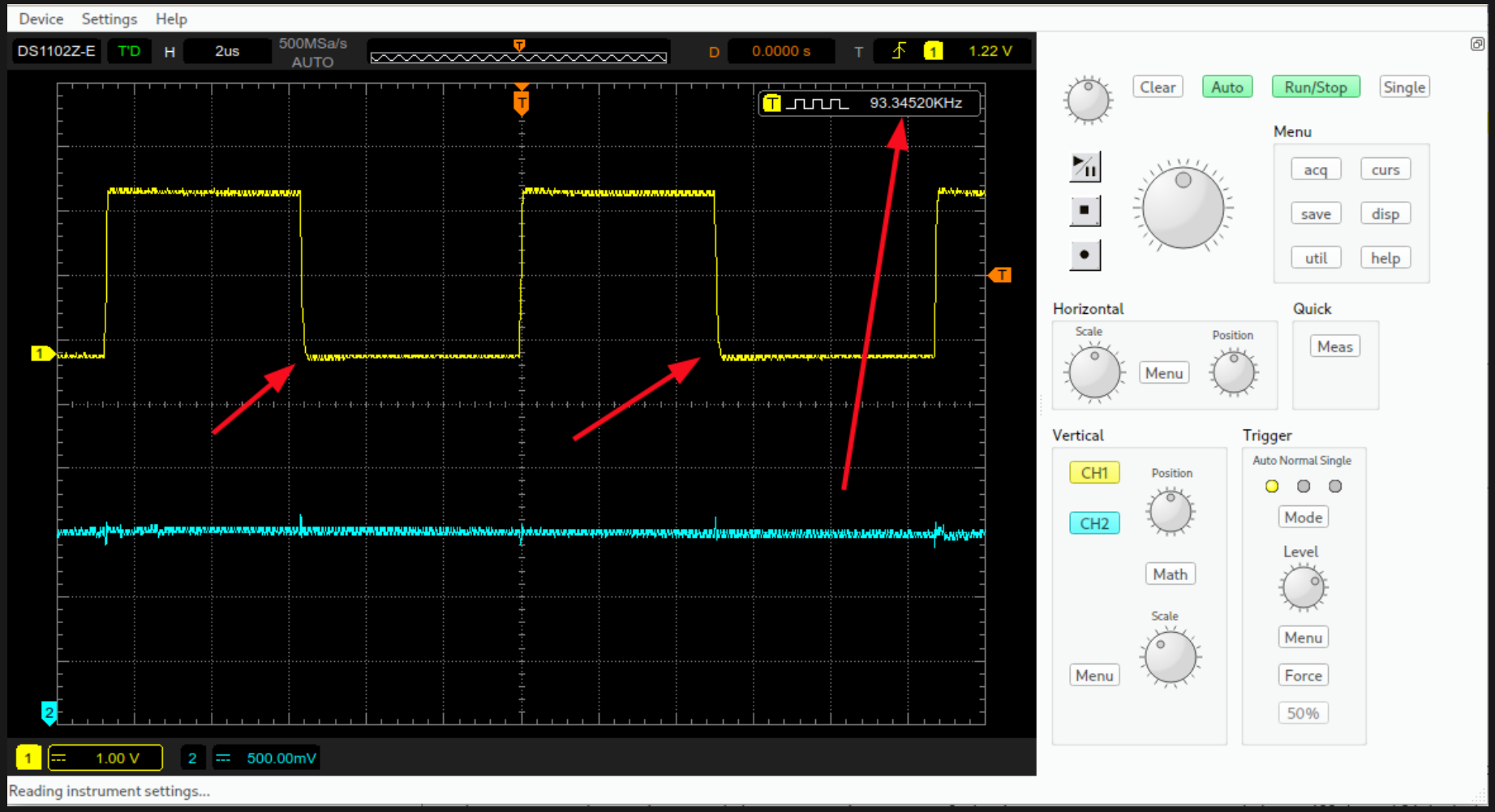
SPEED MICROPYTHON



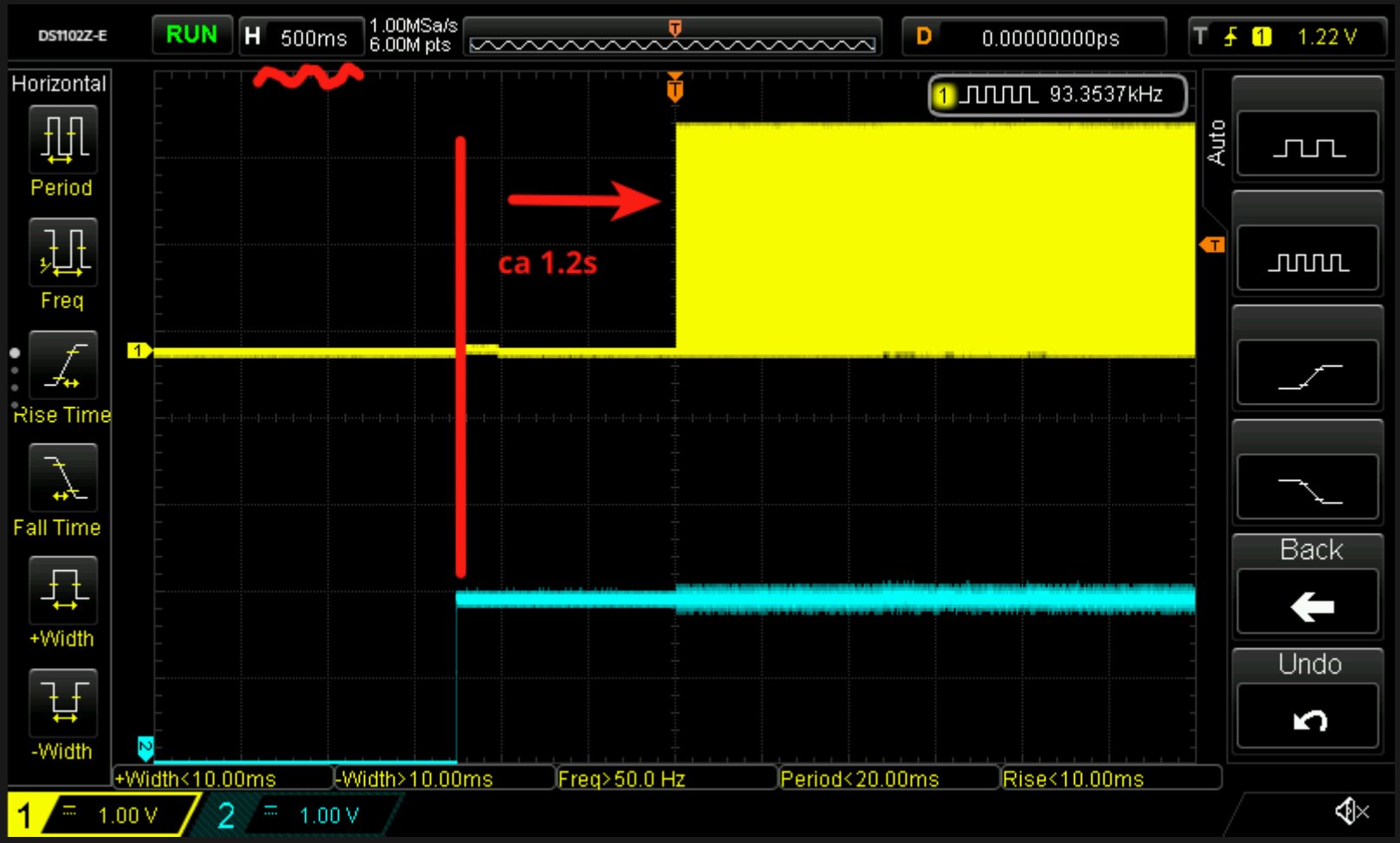
STARTUP MIROPYTHON



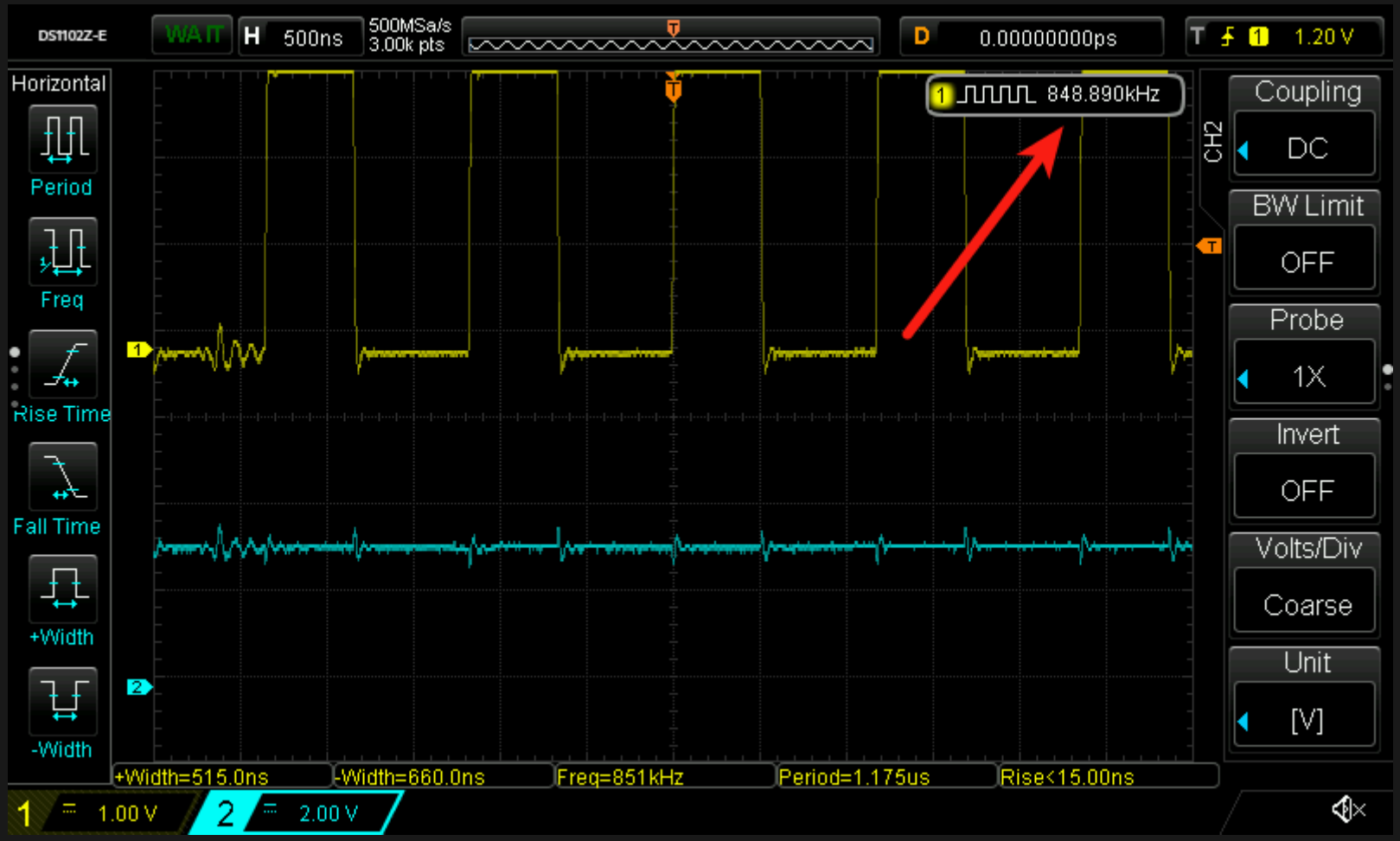
SPEED CIRCUITPYTHON



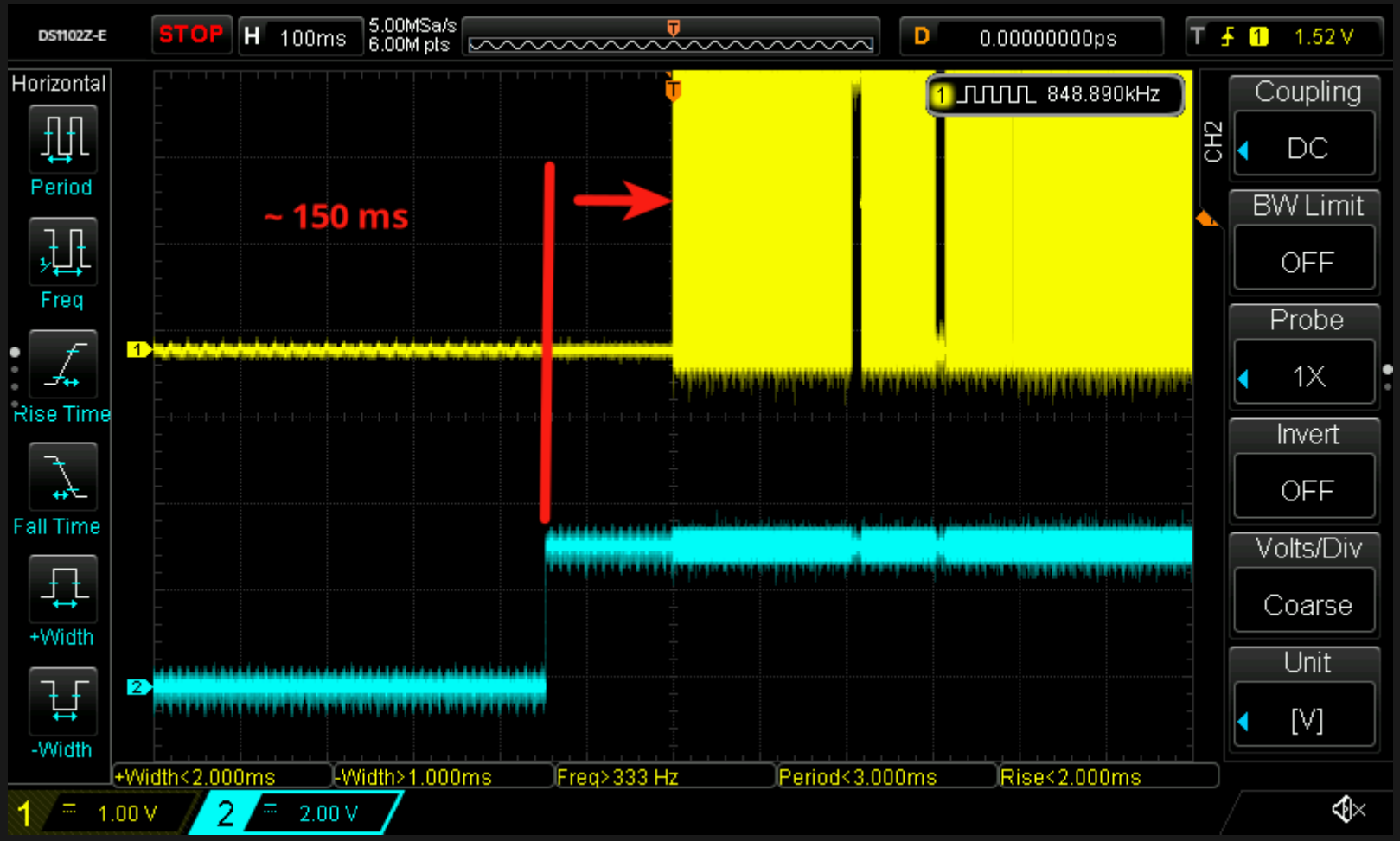
STARTUP CIRCUITPYTHON



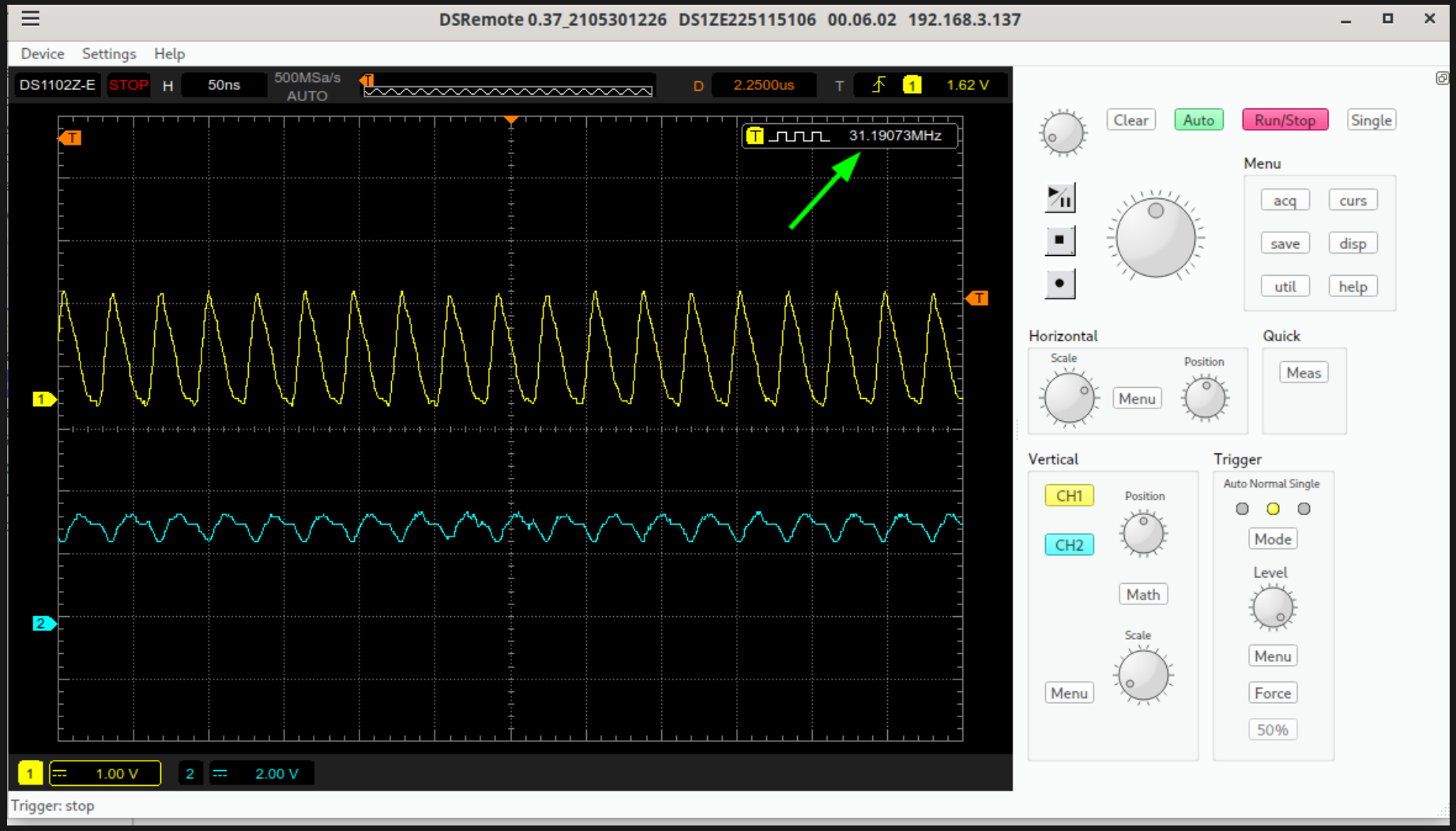
SPEED ARDUINO



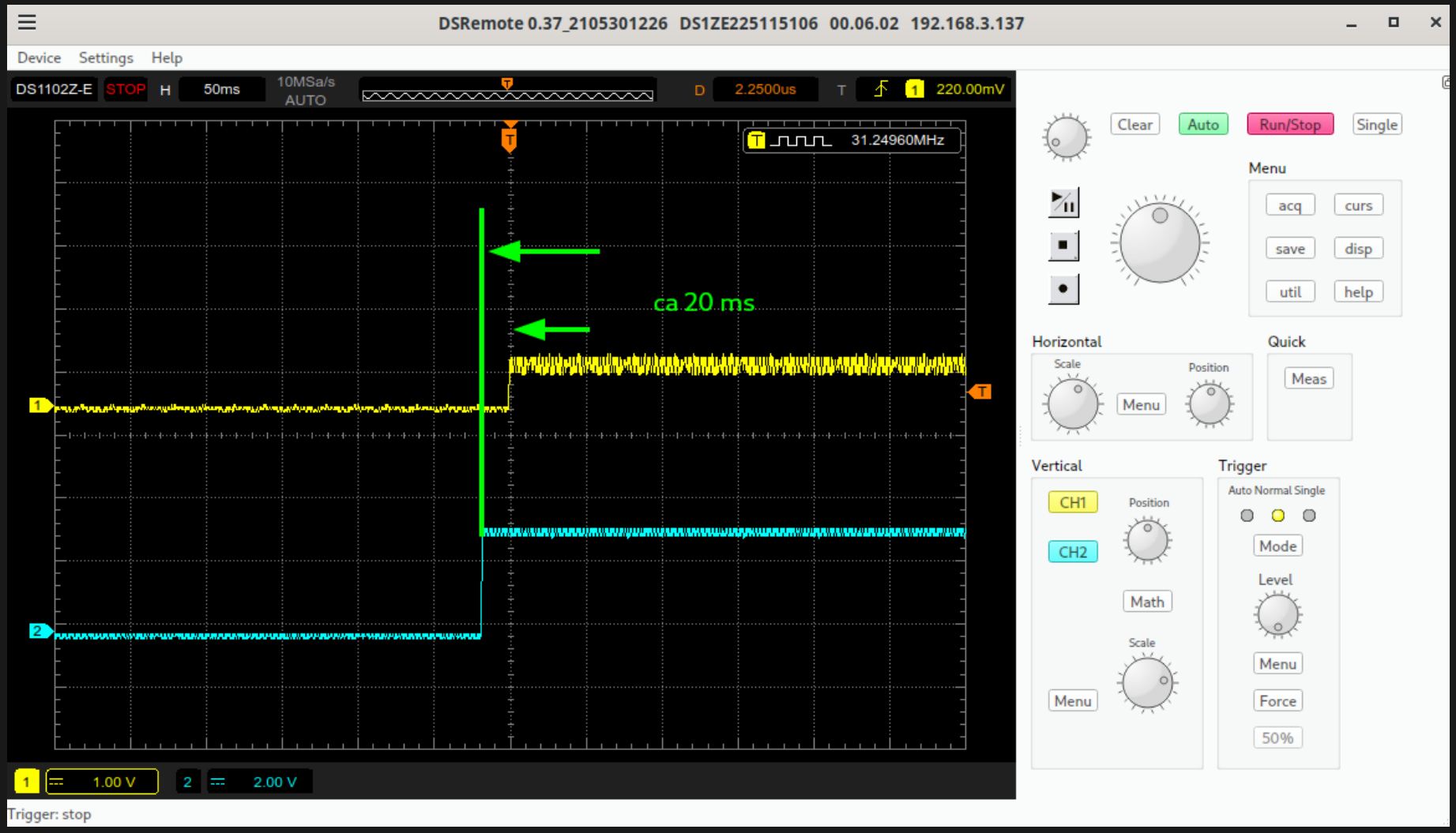
STARTUP ARDUINO



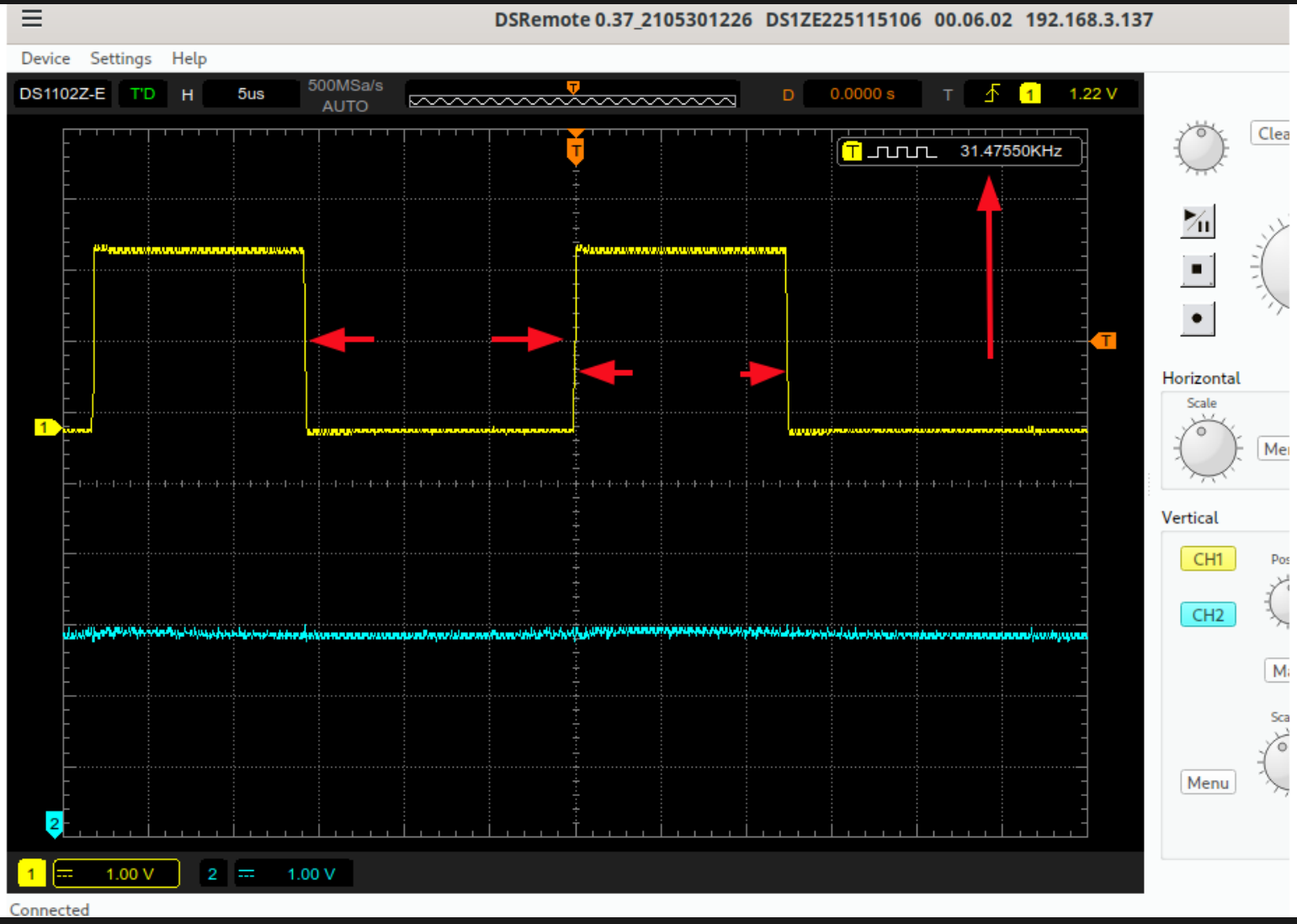
SPEED SDK


DSRemote 0.37_2105301226 DS1ZE225115106 00.06.02 192.168.3.137
Device Settings Help
DS1102Z-E STOP H 50ns 500MSa/s AUTO D 2.2500us T 1 1.62 V
31.19073MHz
1 1.00 V 2 2.00 V
Trigger: stop
Clear Auto Run/Stop Single
Menu
acq curs save disp util help
Horizontal
Scale Menu Position
Quick
Meas
Vertical
CH1 CH2 Position Math Scale Menu
Trigger
Auto Normal Single
Mode Level Menu Force 50%

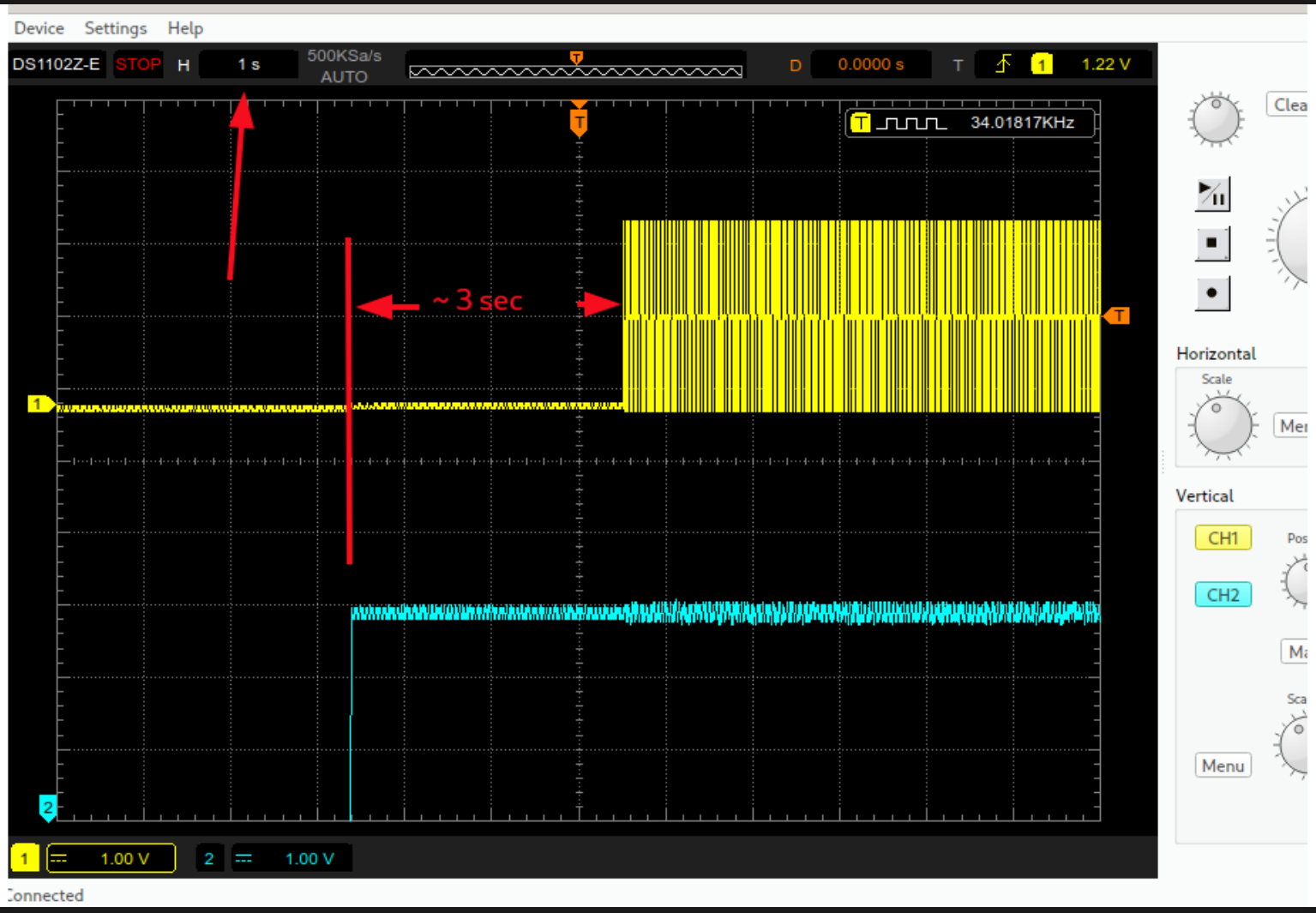
STARTUP SDK


DSRemote 0.37_2105301226 DS1ZE225115106 00.06.02 192.168.3.137
Device Settings Help
DS1102Z-E STOP H 50ms 10MSa/s AUTO D 2.2500us T 1 220.00mV
31.24960MHz
ca 20 ms
1 == 1.00 V 2 == 2.00 V
Trigger: stop
Clear Auto Run/Stop Single
Menu
acq curs save disp util help
Horizontal
Scale Menu Position
Quick
Meas
Vertical
CH1 CH2 Position Math Scale Menu
Trigger
Auto Normal Single
Mode
Level
Menu Force 50%

SPEED MMBASIC



STARTUP MMBASIC



SUMMARY

	Getting started	Execution Speed	Startup Delay
MMBasic	Okay	30 kHz	3000 ms
Micro-Python	Easy	74 kHz	100 ms
Circuit Python	even more easy	90 kHz	1200 ms
Arduino	Easy/Okay	850 kHz	150 ms
SDK	Most effort	30 000 kHz	20 ms

SPEED AN ISSUE ?

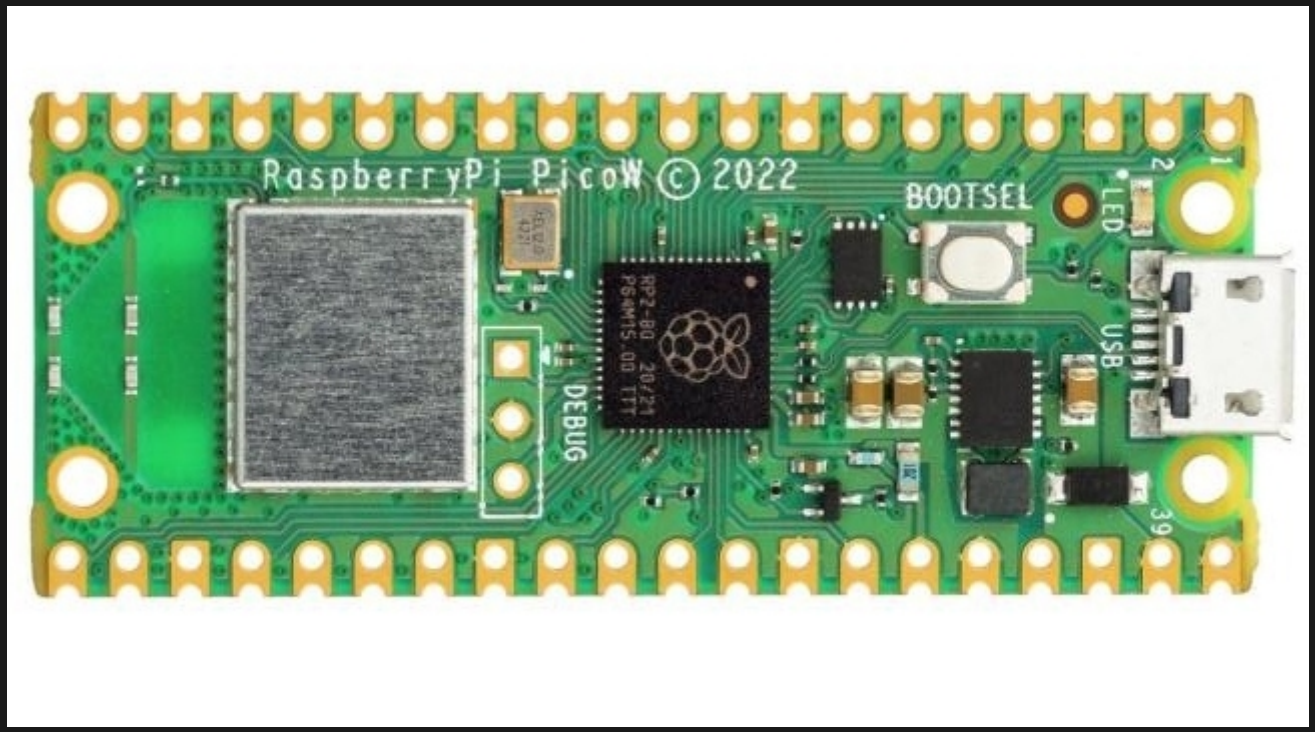
Video

THE PIO

- Programmable IO
- "a Co-Processor", dedicated to programm even new serial "buses"
- offloads work from the CPU, compared to "bit-banging"
- https://www.cnx-software.com/2021/01/27/a-closer-look-at-raspberry-pi-rp2040-programmable-ios-pio/

Unfortunately : Did not make it into this talk ...

2022 : PICO WITH WIFI



EASY WEBSERVER

- Serve Web-Pages
- Control LEDs
- Starting with PicoZero, even more simple

DEMO 2

WHAT WE (HOPEFULLY) SAW

PICO-W AS SERVER

ADDITIONAL HW

- ePaper => Python
- Round LCD w Gyroscope => Python
- Grove Breakout Board => Python
- Small LCD with Key => Mandelbrot ! => SDK

DEMO 3

WHAT WE (HOPEFULLY) SAW

PICO EPAPER

ROUND LCD

GROVE-SENSORS

CONCLUSION FOR PICO

5 Different Dev-Environments

- C/C++ - SDK via cmake and U2F-Filesystem
- Micro-Python with USB-serial / Thonny
- Circuit-Python with real filesystem
- Arduino - IDE
- MMBasic

Know the limits and choose yourself

MEETUP ZUERICH

Every first monday in the month in Zuerich : here

MEETUP BRUNNEN

every four weeks , wednesday or thursday in Brunnen : here

RESOURCES

- this talk git-repo (branch gh_pages)
- this talk as slides

PYTHON

- Micropython Pi Pico: Info U2F-Image
- Circuitpython Pi Pico: Info U2F-Image
- Getting started with Raspberry Pi Pico and Micro-Python Book

BASIC

- Editor for Basic on Pico : https://www.c-com.com.au/M
- Basic for the PICO : https://geoffg.net/picomite.html
- The Description : https://geoffg.net/Downloads/picomite/PicoMite_Use
- The Firmware : https://geoffg.net/Downloads/picomite/PicoMite_Firm
- The Source-Code : https://github.com/UKTailwind/Pic
- https://www.heise.de/ratgeber/Programmieren-mit-d Raspberry-Pi-Pico-Back-to-BASIC-7461038.html

SDK

- C/C++ - SDK for Raspberry Pi Pico: Info and the GIT-Repository

PYTHON / PICO / WEB

- A WEB-Server with Micro-Dot : http://www.doctormonk.com/2022/09/a-better-web-server-for-raspberry-pi.html
- With PHEW : https://www.youtube.com/watch?v=0sPPxIq4hg8
- PHEW: https://pypi.org/project/micropython-phew/

FREE MAGAZINES

- Hackspace Magazines by Raspberry Pi Foundation : https://hackspace.raspberrypi.com/issues

THE END