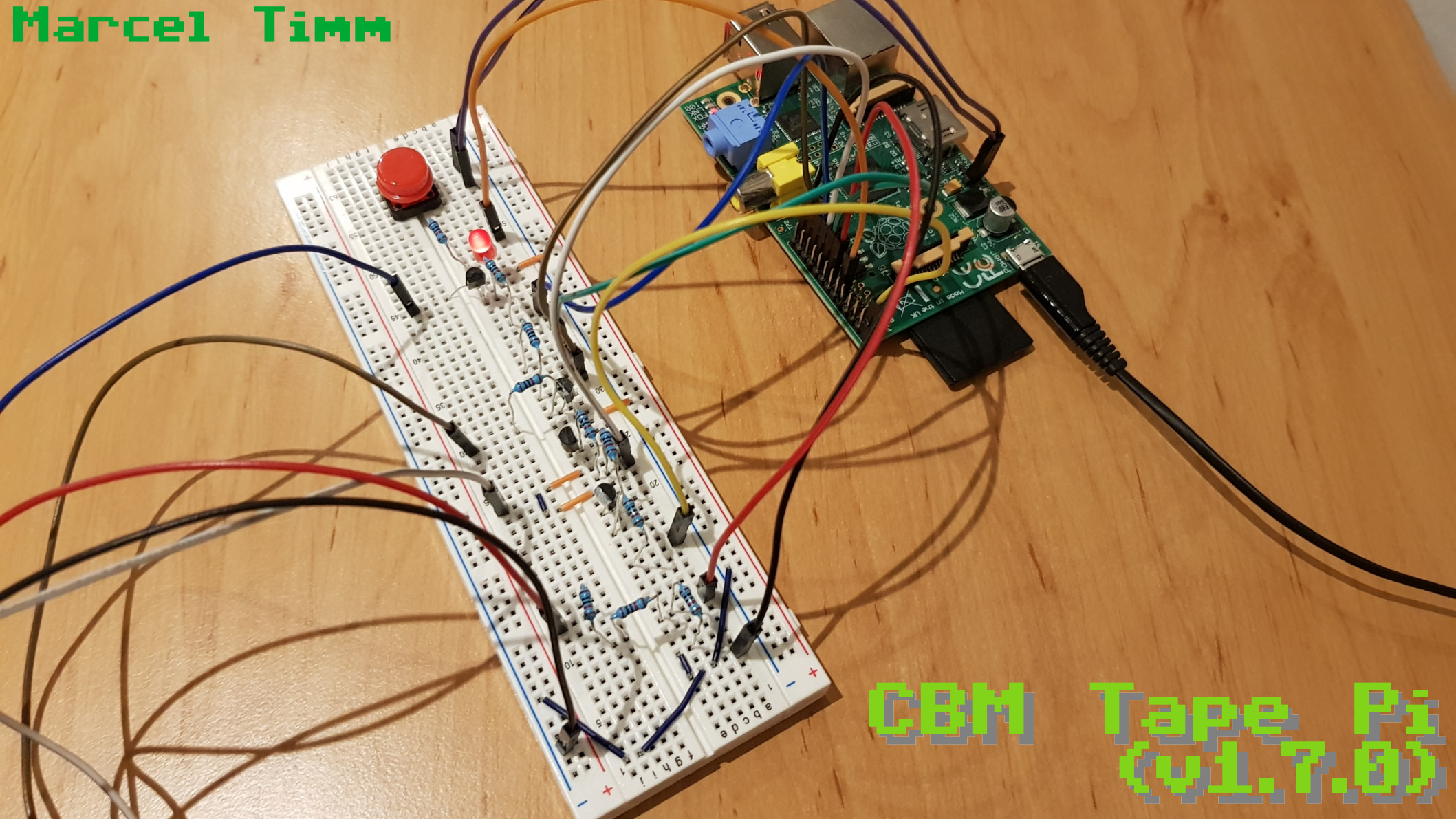

Marcel Timm
CBM Tape Pi
(v1.7.0)

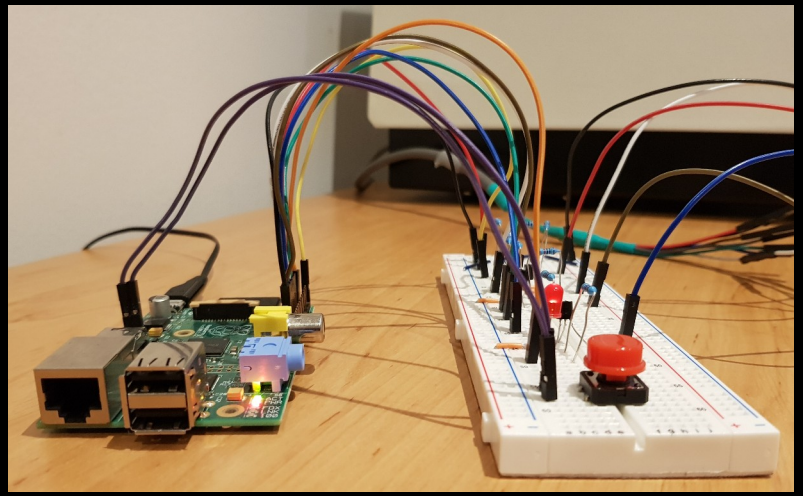
INTRODUCTION

INTRODUCTION

Marcel Timm

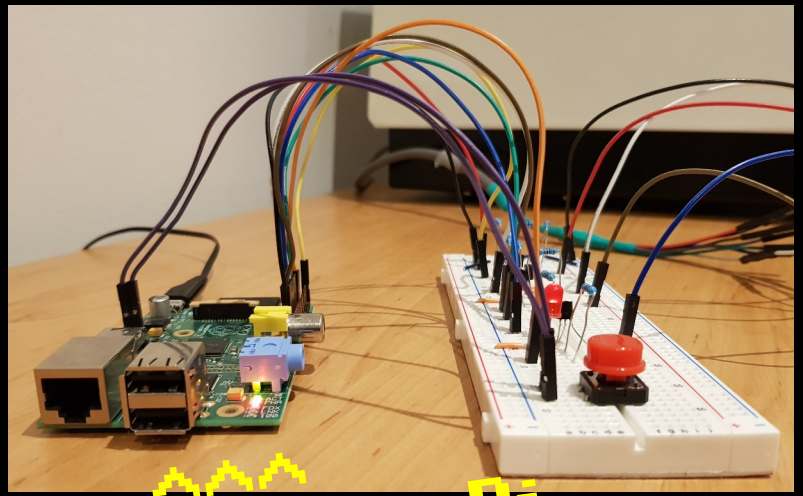
INTRODUCTION

Marcel Timm



INTRODUCTION

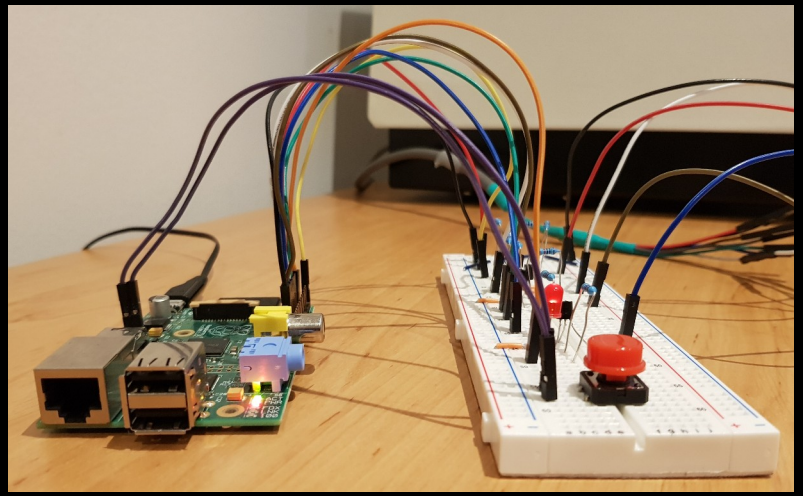
Marcel Timm



Raspberry Pi

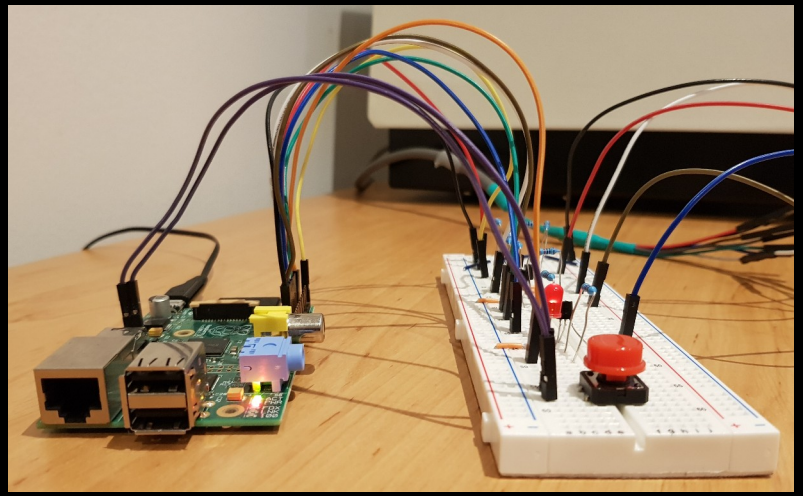
INTRODUCTION

Marcel Timm



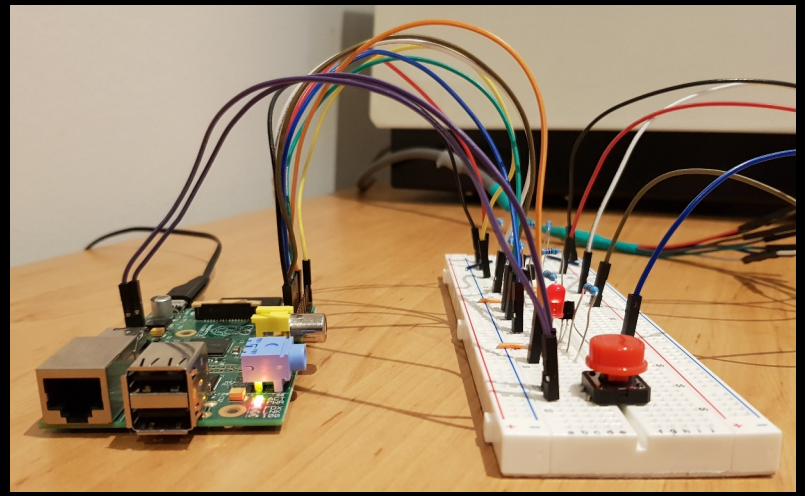
INTRODUCTION

Marcel Timm



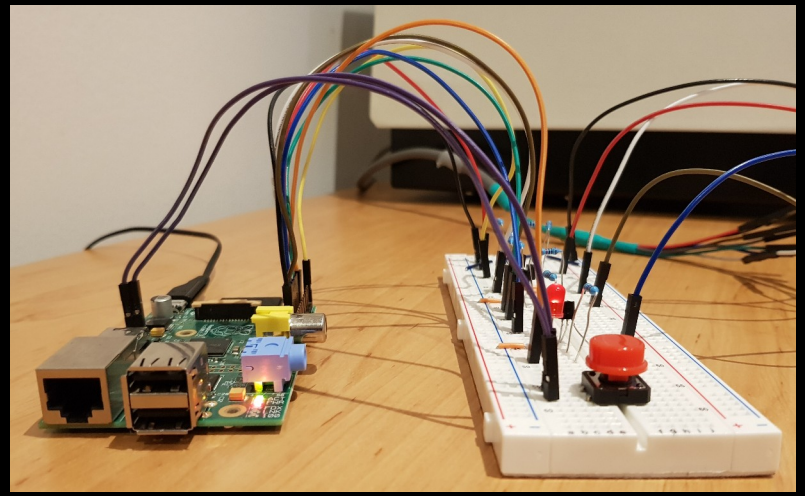
INTRODUCTION

Marcel Timm



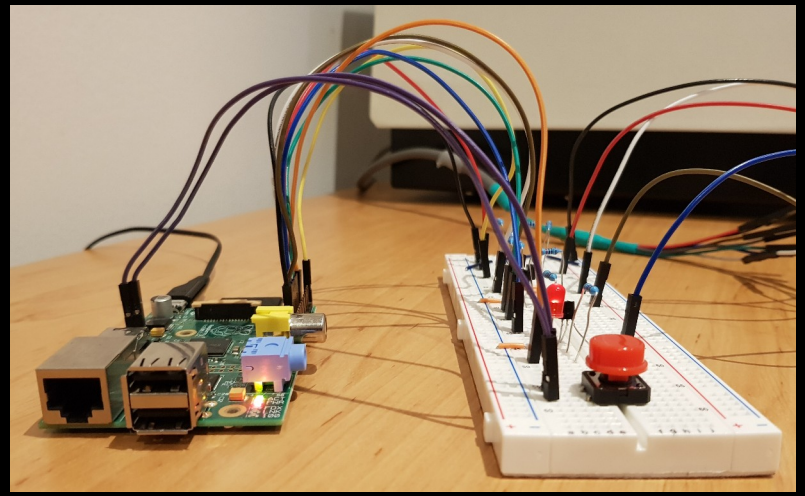
INTRODUCTION

Marcel Timm



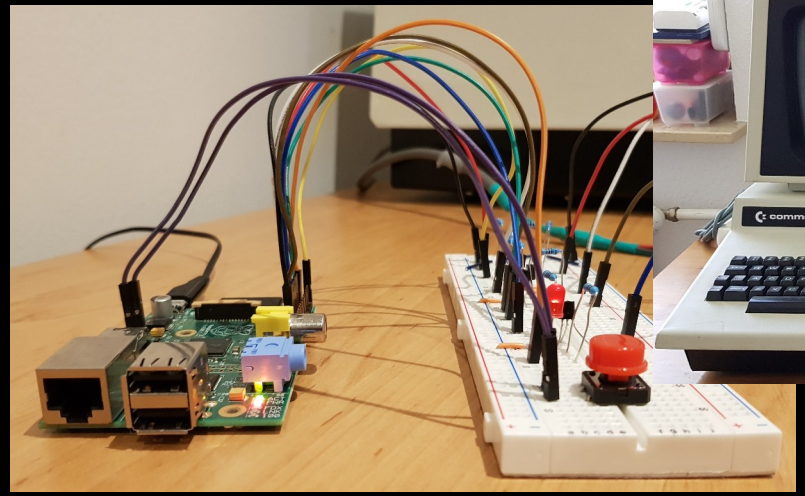
INTRODUCTION

Marcel Timm



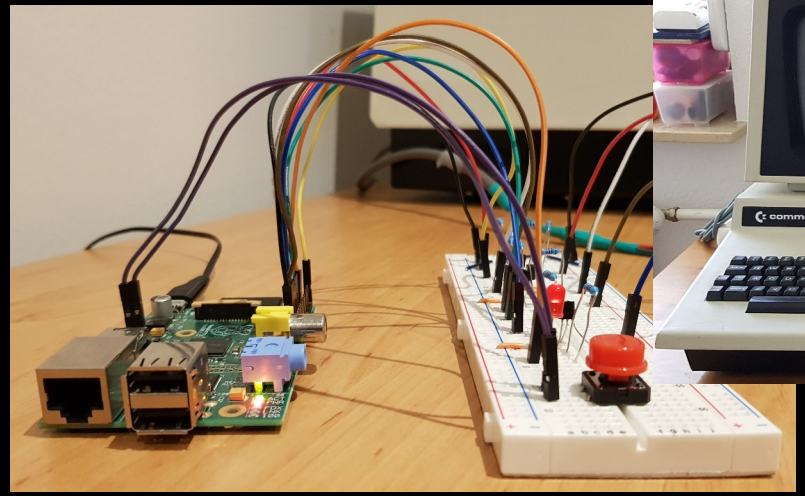
INTRODUCTION

Marcel Timm



INTRODUCTION

Marcel Timm



Use an SD card!

"ALTERNATIVES"

"ALTERNATIVES"

"ALTERNATIVES"

* SD2IEC
* 1541 Ultimate II+
* EasyFlash 3
* petSD
* Tapuino
* Tapecart
* …

"ALTERNATIVES"

* SD2IEC
* 1541 Ultimate II+
* EasyFlash 3
* petSD
* Tapuino
* Tapecart
* …

> WHY CBM TAPE PI? <

WHY CBM TAPE PI?

WHY CBM TAPE PI?

Reasons for me:

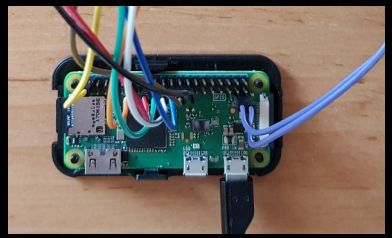
- send own cross-developed PRGs to PET/CBM fast
- creating something closer to the hardware than C# and JavaScript
- learning Bare Metal C
- improving my 6502 assembler and Commodore knowledge
- creating a small circuit

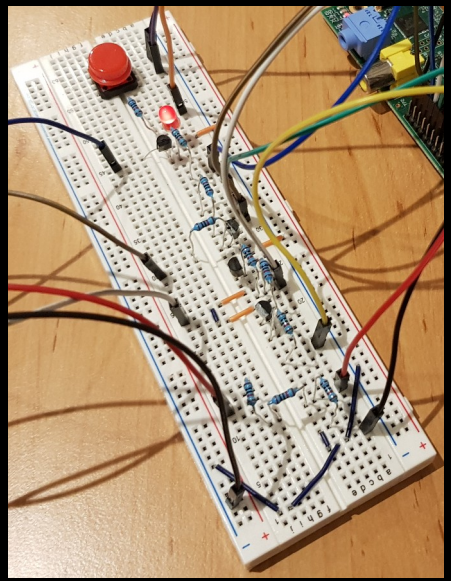
WHY CBM TAPE PI?

Reasons for YOU (hopefully):

- cheap solution
- parts readily available
- easy to build & set up
- occupies tape port, only
- high compatibility
- fast
- open source

PARTS, COSTS, AVAILABILITY





- Raspberry Pi 1, 2, 3, 0
- simple interfacing circuit
- discrete components, only

=>

- no ordering from far away
- no taxes
- no long waiting

BUILDING CBM TAPE PI

- datassette port in use, all other ports free
- you can use a breadboard for the interface circuit

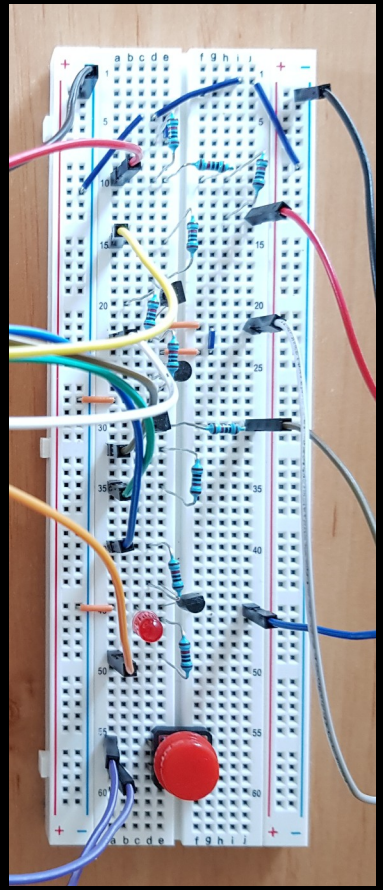
  => only 5 wires to solder
- less error prone:

  * solder on stripboard

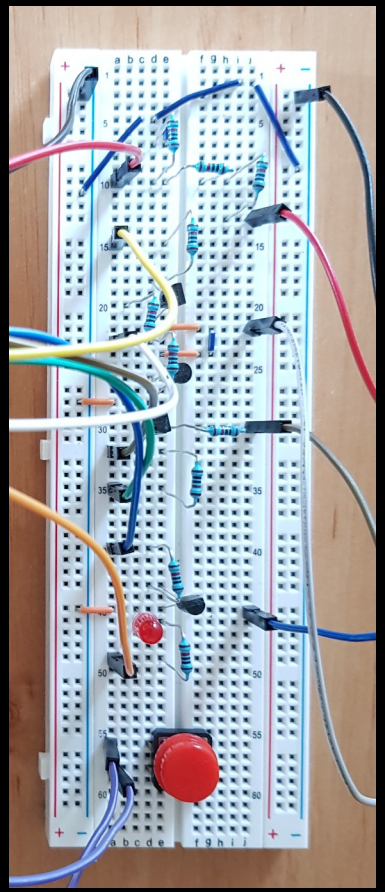
  * or use PCB

				N.C.	
6/F	5/E	4/D	3/C	2/B	1/A
Sense	Write	Read	Motor		GND

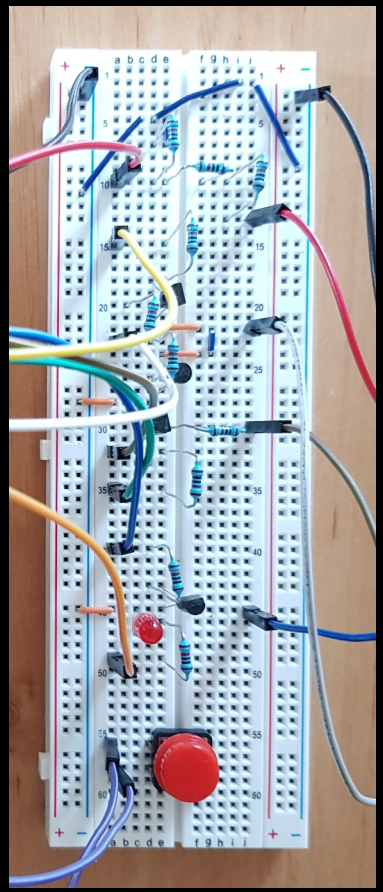
BUILDING CBM TAPE PI



BUILDING CBM TAPE PI



BUILDING CBM TAPE PI



COMPATIBILITY

COMPATIBILITY

- C64

COMPATIBILITY

- C64
- VIC 20

COMPATIBILITY

- C64
- VIC 20
- PET/CBM 2001, 3032, 4032, 8032, …

COMPATIBILITY

- C64
- VIC 20
- PET/CBM 2001, 3032, 4032, 8032, …
    * BASIC v4

COMPATIBILITY

- C64
- VIC 20
- PET/CBM 2001, 3032, 4032, 8032, …
    * BASIC v4
    * BASIC v2

COMPATIBILITY

- C64
- VIC 20
- PET/CBM 2001, 3032, 4032, 8032, …
    - * BASIC v4
    - * BASIC v2
    - * BASIC v1

OPERATION MODES

* COMPATIBILITY MODE:

* FAST MODE:

OPERATION MODES

* COMPATIBILITY MODE:

- no extra software on Commodore needed, uses Kernal routines

* FAST MODE:

OPERATION MODES

* COMPATIBILITY MODE:

- no extra software on Commodore needed, uses Kernal routines

* FAST MODE:

- needs PRG file to be installed

OPERATION MODES

* COMPATIBILITY MODE:

- no extra software on Commodore needed, uses Kernal routines
- controlled via SAVE & LOAD

* FAST MODE:

- needs PRG file to be installed

OPERATION MODES

* COMPATIBILITY MODE:

- no extra software on Commodore needed, uses Kernal routines
- controlled via SAVE & LOAD

* FAST MODE:

- needs PRG file to be installed
- controlled via extra commands (wedge)

OPERATION MODES

* COMPATIBILITY MODE:

- no extra software on Commodore needed, uses Kernal routines
- controlled via SAVE & LOAD
- just as slow as the real datassette

* FAST MODE:

- needs PRG file to be installed
- controlled via extra commands (wedge)

OPERATION MODES

* COMPATIBILITY MODE:

- no extra software on Commodore needed, uses Kernal routines
- controlled via SAVE & LOAD
- just as slow as the real datassette

* FAST MODE:

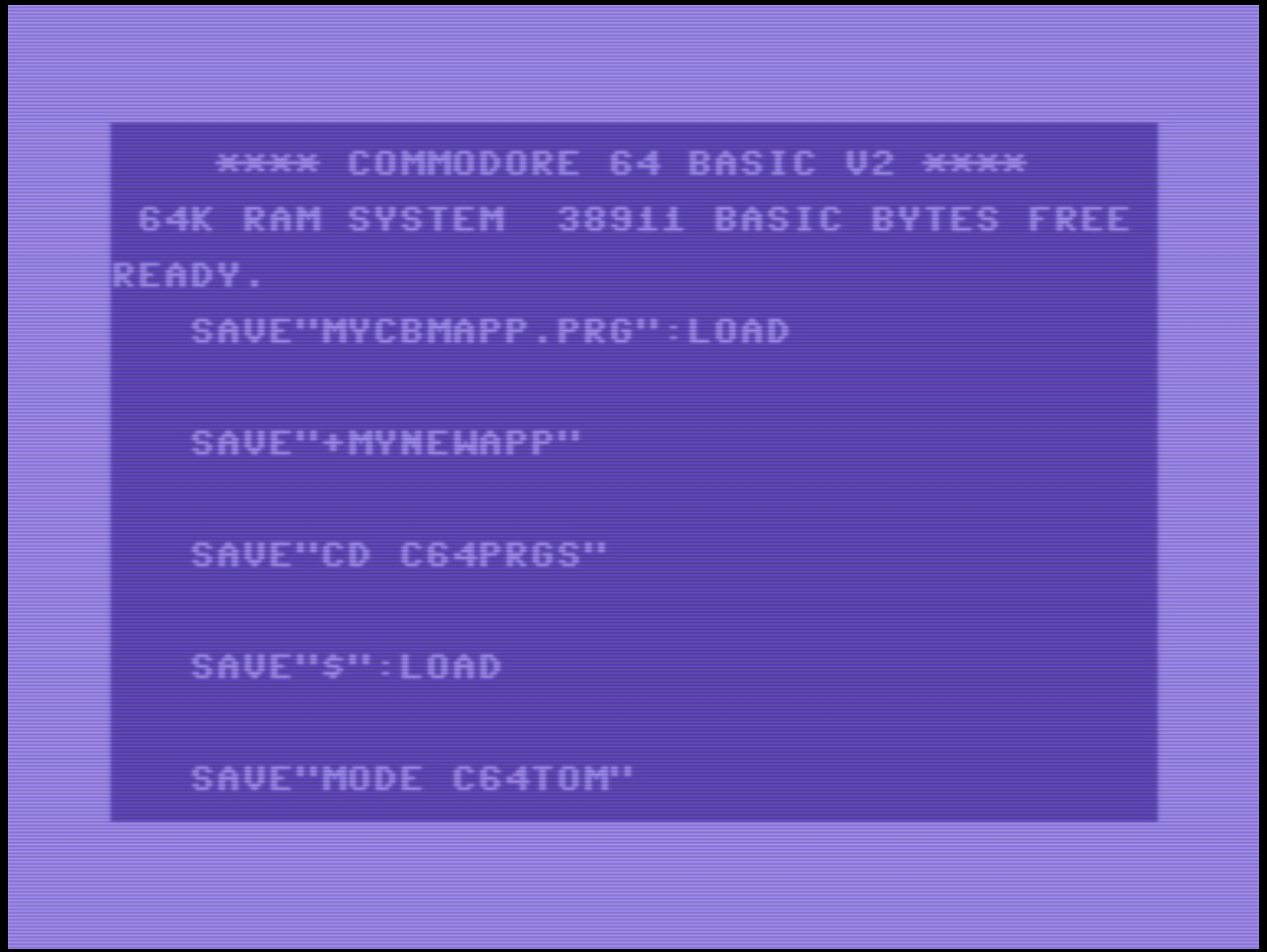
- needs PRG file to be installed
- controlled via extra commands (wedge)
- ~2.5kB/s (e.g. faster than tape or 1541)

OPERATION MODES

* COMPATIBILITY MODE:
  - Not tested: Might work with C16, etc.
- no extra software on Commodore needed, uses Kernal routines
- controlled via SAVE & LOAD
- just as slow as the real datassette

* FAST MODE:
- needs PRG file to be installed
- controlled via extra commands (wedge)
- ~2.5kB/s (e.g. faster than tape or 1541)

EXAMPLES: COMPATIBILITY MODE



EXAMPLES: COMPATIBILITY MODE

Loading a PRG from SD card:

EXAMPLES: COMPATIBILITY MODE

Loading a PRG from SD card:

Saving from RAM to SD card:

EXAMPLES: COMPATIBILITY MODE

Loading a PRG from SD card:

```
SAVE"MYCBMAPP.PRG":LOAD
```

Saving from RAM to SD card:

```
SAVE"+MYNEWAPP"
```

Changing current directory:

```
SAVE"CD C64PRGS"

SAVE"$":LOAD

SAVE"MODE C64TOM"
```

EXAMPLES: COMPATIBILITY MODE

Loading a PRG from SD card:

Saving from RAM to SD card:

Changing current directory:

Loading directory listing:

EXAMPLES: COMPATIBILITY MODE

Loading a PRG from SD card:

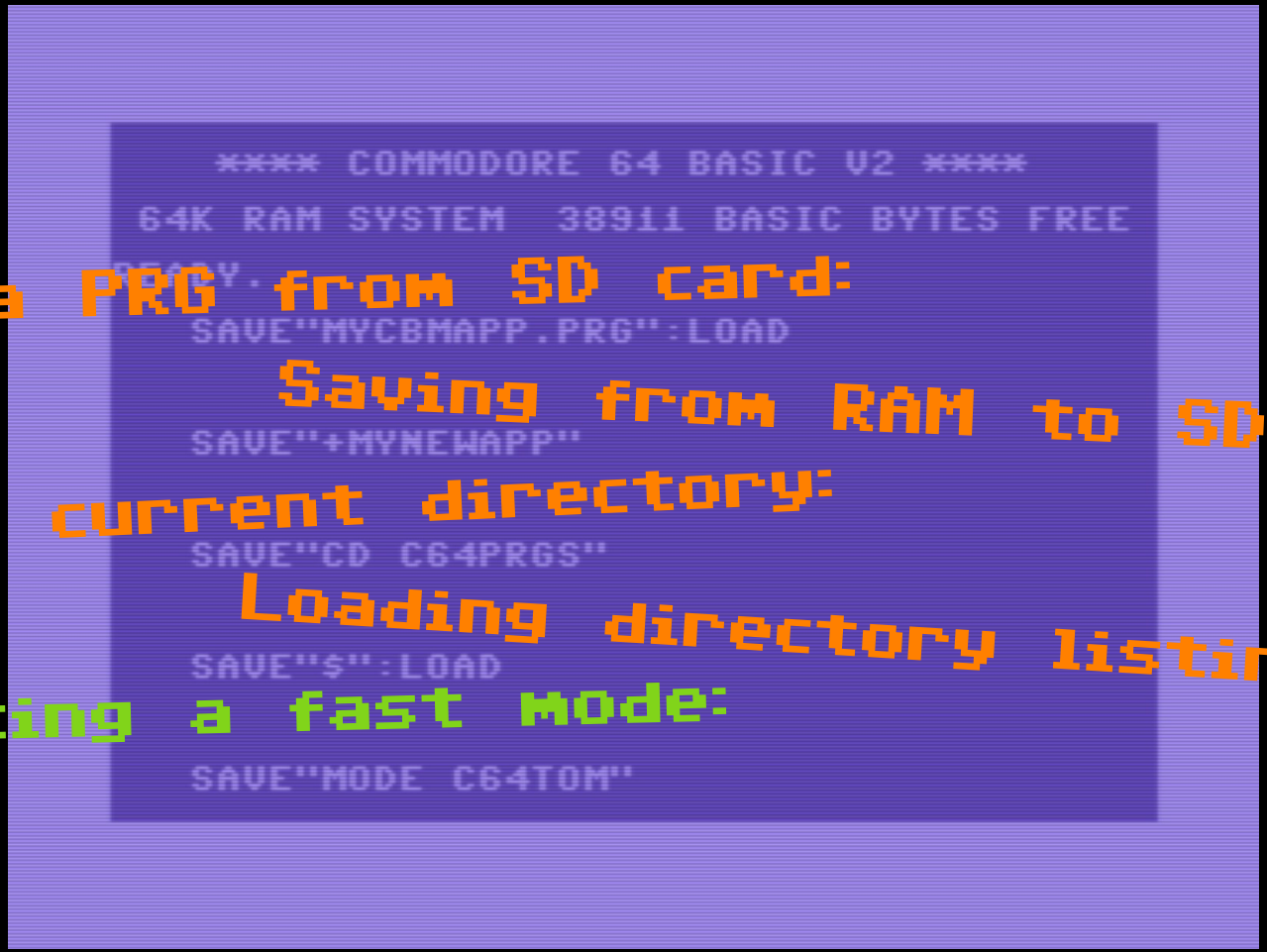
Saving from RAM to SD card:

Changing current directory:

Loading directory listing:

Selecting a fast mode:

EXAMPLES: COMPATIBILITY MODE



Loading a PRG from SD card:

Saving from RAM to SD card:

Changing current directory:

Loading directory listing:

Selecting a fast mode:

ENABLING (A) FAST MODE: Options:

ENABLING (A) FAST MODE: Options:

Enter in compatibility mode:

SAVE"MODE <put mode title, here>"

ENABLING (A) FAST MODE: Options:

Enter in compatibility mode:

SAVE"MODE <put mode title, here>"

CBM/PET:
* selects BASIC version (1, 2, 4)

ENABLING (A) FAST MODE: Options:

Enter in compatibility mode:

SAVE"MODE <put mode title, here>"

CBM/PET:

* selects BASIC version (1, 2, 4)

* installs wedge to tape buffers or TOM (top of BASIC memory)

ENABLING (A) FAST MODE: Options:

Enter in compatibility mode:

SAVE"MODE <put mode title, here>"

CBM/PET:

* selects BASIC version (1, 2, 4)

* installs wedge to tape buffers or TOM (top of BASIC memory)

VIC 20:

* always installs to TOM

ENABLING (A) FAST MODE: Options:

Enter in compatibility mode:

SAVE"MODE <put mode title, here>"

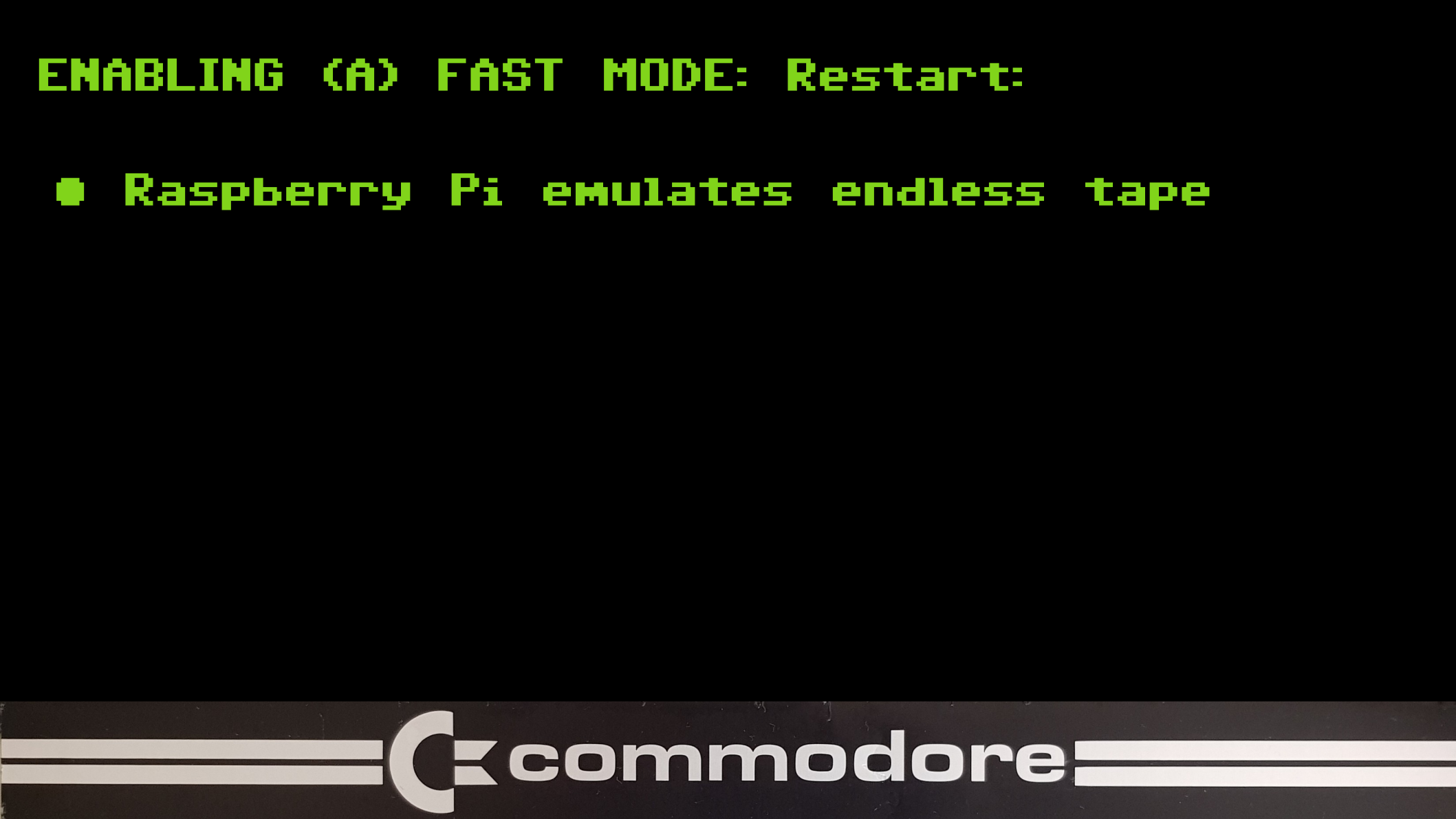
CBM/PET:
* selects BASIC version (1, 2, 4)

* installs wedge to tape buffers or TOM (top of BASIC memory)

VIC 20:
* always installs to TOM

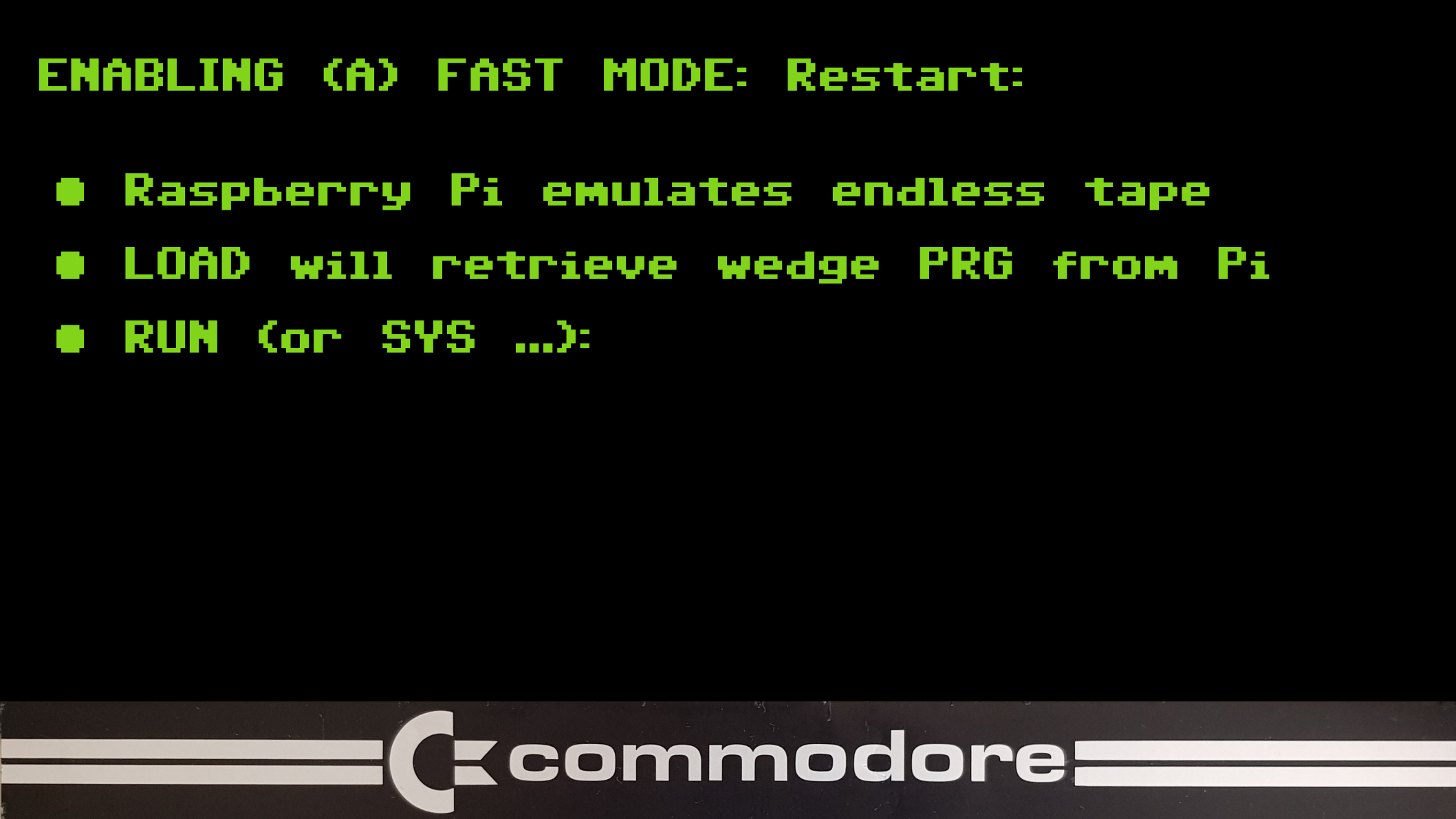
C64:
* installs command extension to TOM or to top of free high memory (right before address $D000)


ENABLING (A) FAST MODE: Restart:
commodore


ENABLING (A) FAST MODE: Restart:
● Raspberry Pi emulates endless tape
commodore

ENABLING (A) FAST MODE: Restart:

- Raspberry Pi emulates endless tape
- LOAD will retrieve wedge PRG from Pi

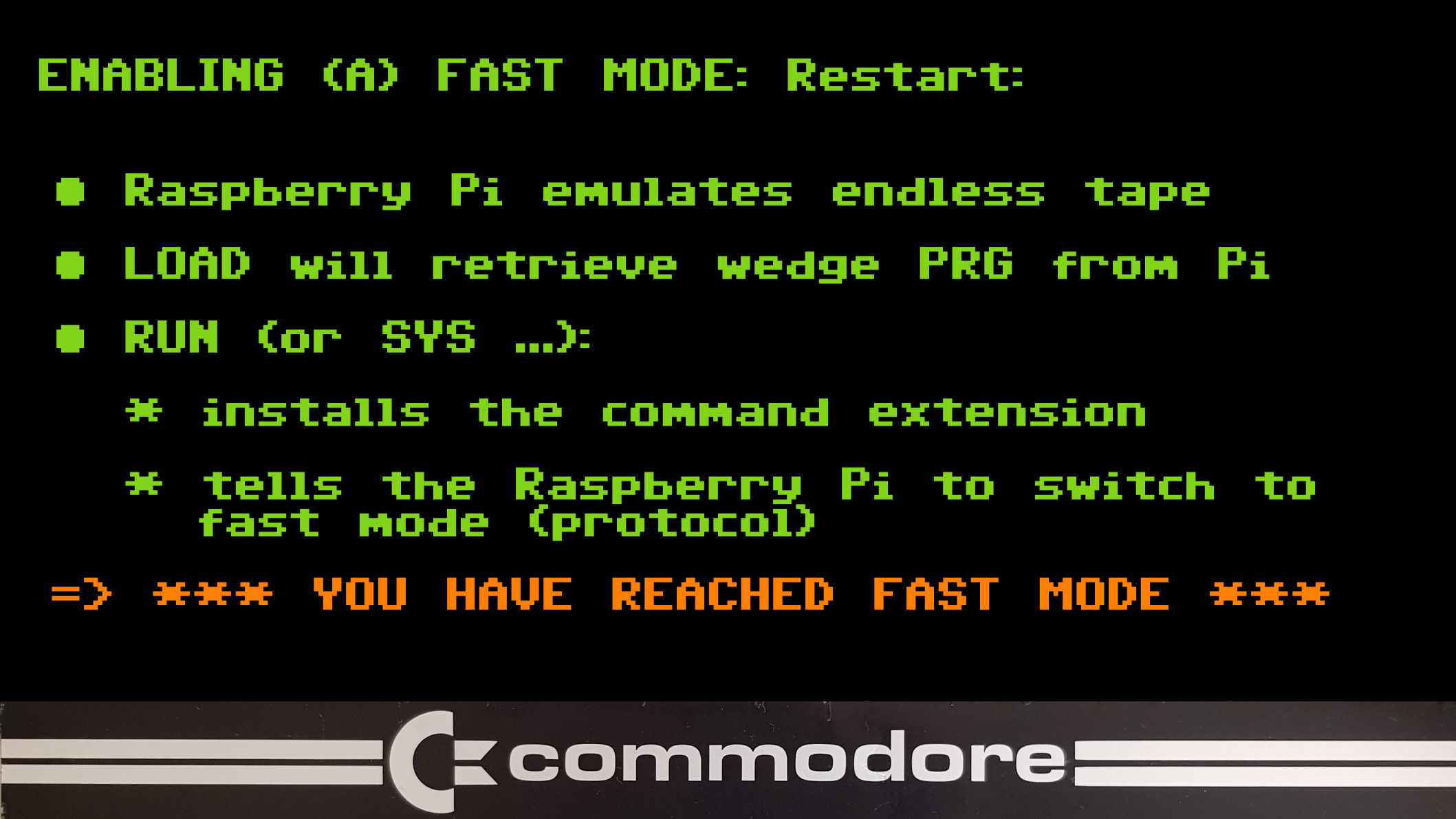

ENABLING (A) FAST MODE: Restart:
● Raspberry Pi emulates endless tape
● LOAD will retrieve wedge PRG from Pi
● RUN (or SYS ...):
commodore

ENABLING (A) FAST MODE: Restart:

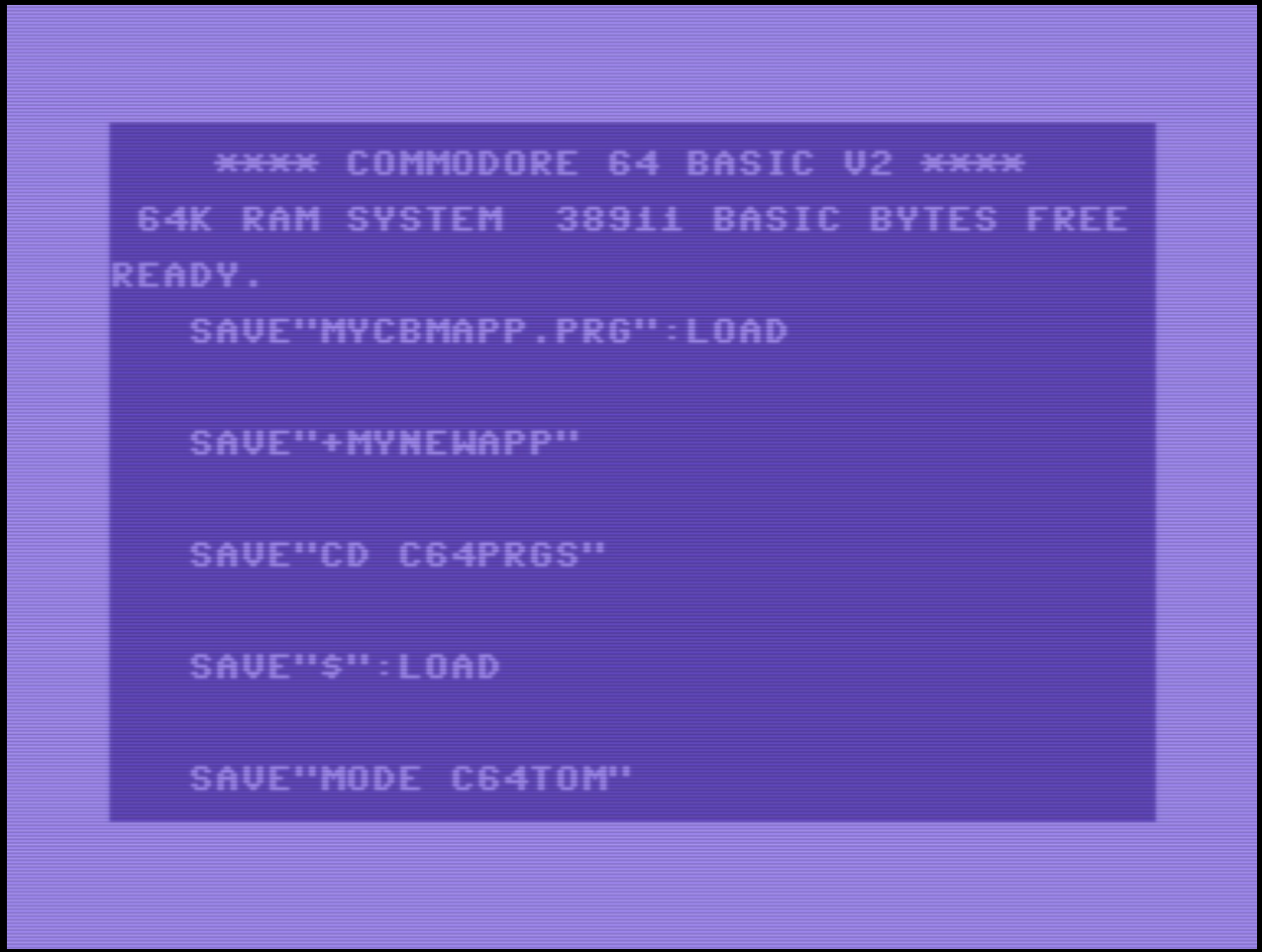
- Raspberry Pi emulates endless tape
- LOAD will retrieve wedge PRG from Pi
- RUN (or SYS …):
  * installs the command extension

ENABLING (A) FAST MODE: Restart:

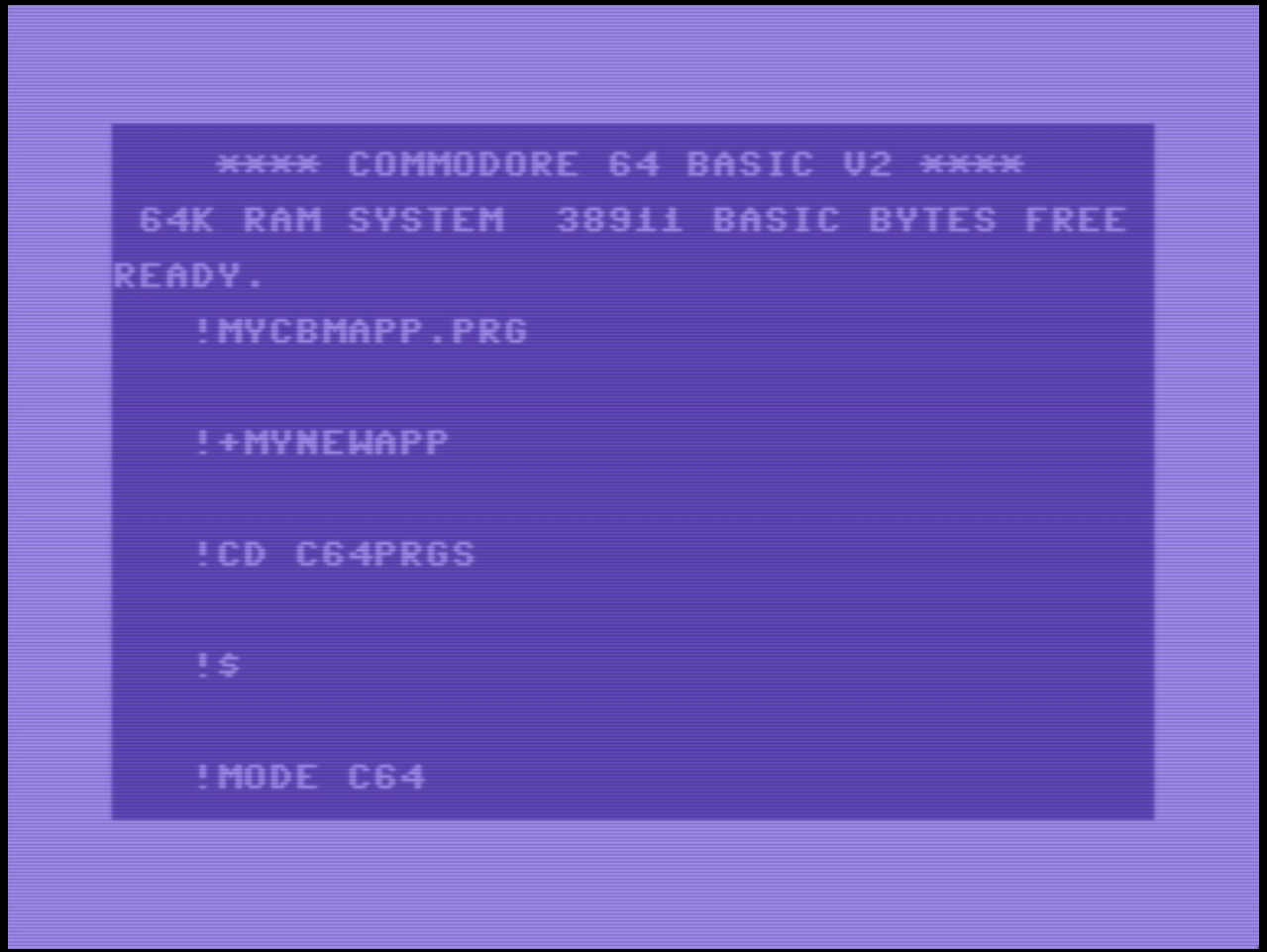
- Raspberry Pi emulates endless tape
- LOAD will retrieve wedge PRG from Pi
- RUN (or SYS …):
  - * installs the command extension
  - * tells the Raspberry Pi to switch to fast mode (protocol)


ENABLING (A) FAST MODE: Restart:
● Raspberry Pi emulates endless tape
● LOAD will retrieve wedge PRG from Pi
● RUN (or SYS ...):
* installs the command extension
* tells the Raspberry Pi to switch to fast mode (protocol)
=> *** YOU HAVE REACHED FAST MODE ***
commodore

EXAMPLES: FAST MODE



EXAMPLES: FAST MODE



EXAMPLES: FAST MODE

Loading a PRG from SD card:

EXAMPLES: FAST MODE

Loading a PRG from SD card:

Saving from RAM to SD card:

EXAMPLES: FAST MODE

Loading a PRG from SD card:

Saving from RAM to SD card:

Changing current directory:

EXAMPLES: FAST MODE

EXAMPLES: FAST MODE

Loading a PRG from SD card:

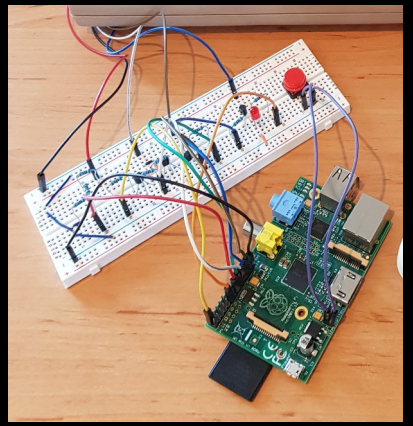
Saving from RAM to SD card:

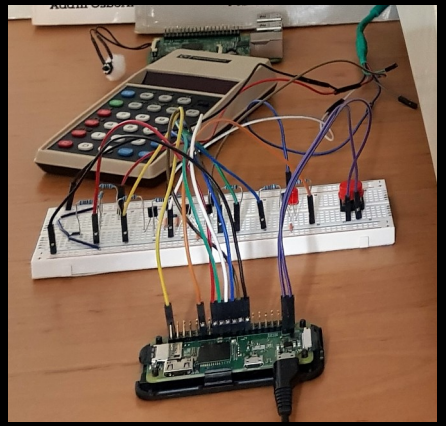
Changing current directory:

Loading directory listing:

Selecting another mode:

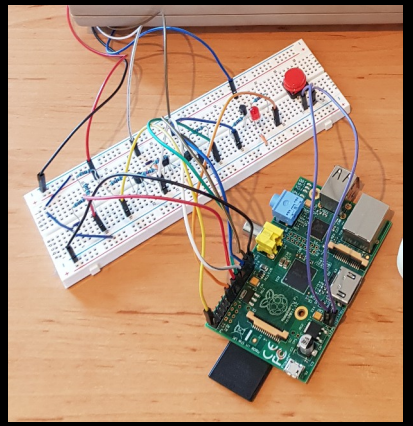
THE FUTURE

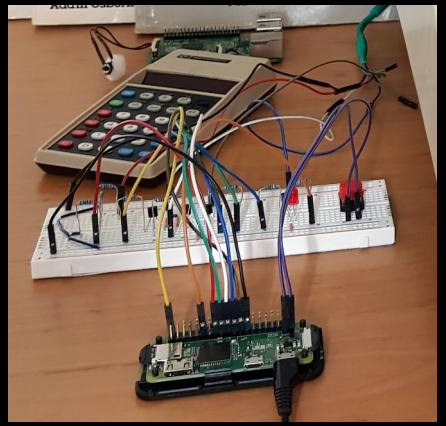




THE FUTURE

Small enhancements:

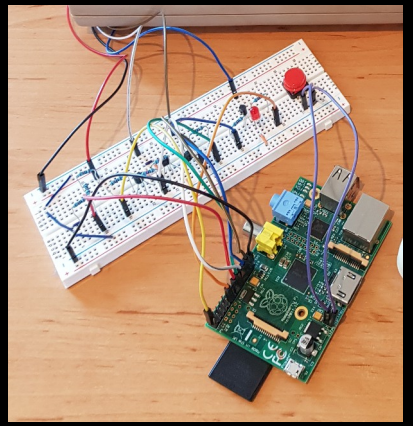


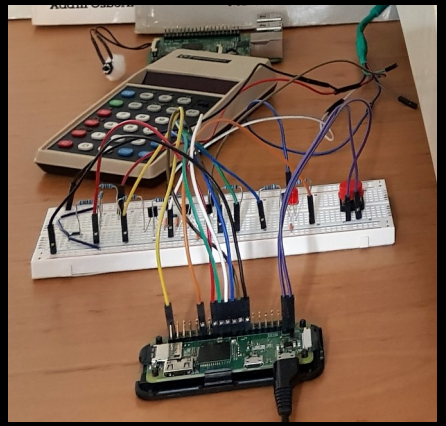


THE FUTURE

Small enhancements:

- remove 8+3 limitation

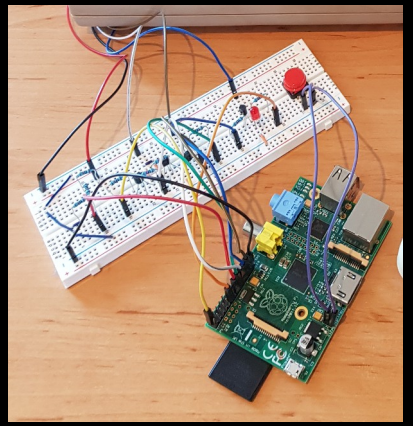


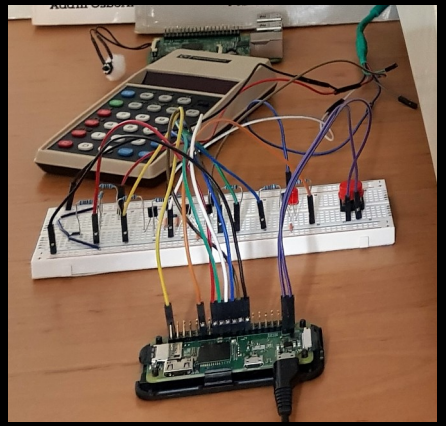


THE FUTURE

Small enhancements:

- remove 8+3 limitation
- !$ without overwrite

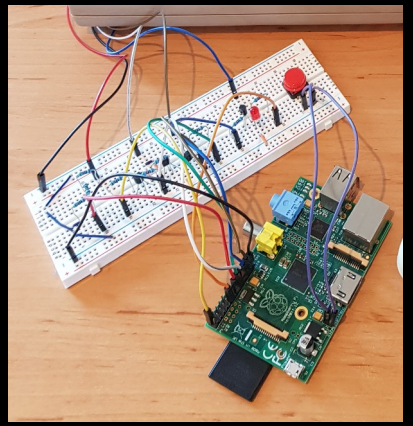


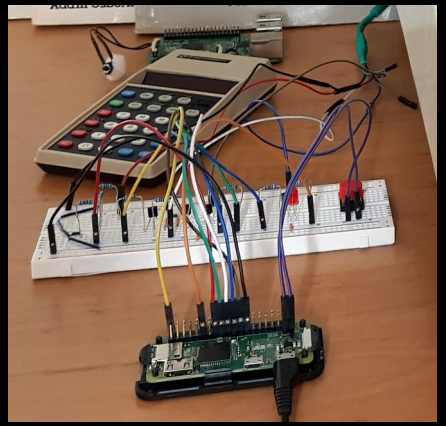


THE FUTURE

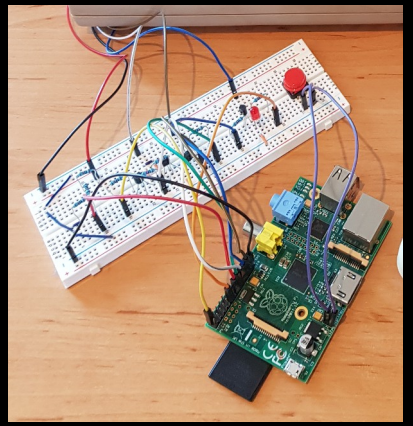
Small enhancements:

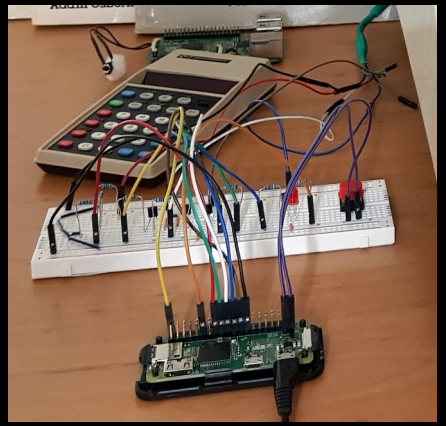
- remove 8+3 limitation
- !$ without overwrite
- let !$ show address & size of PRGs





THE FUTURE

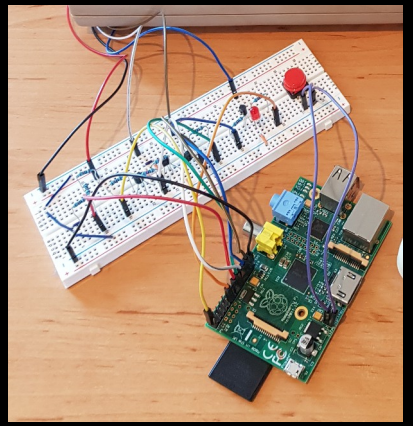


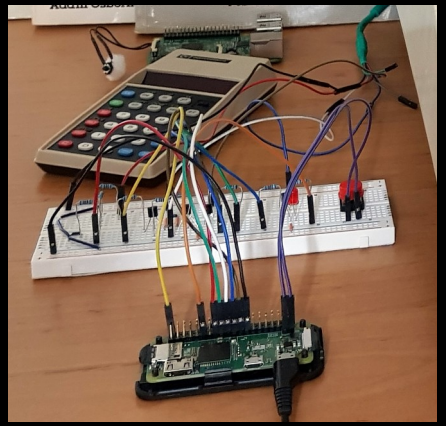


Small enhancements:

- remove 8+3 limitation
- !$ without overwrite
- let !$ show address & size of PRGs
- increase loading speed of wedge PRGs
  (by making them smaller)

THE FUTURE

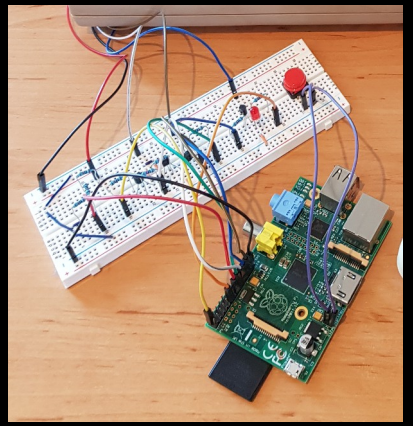


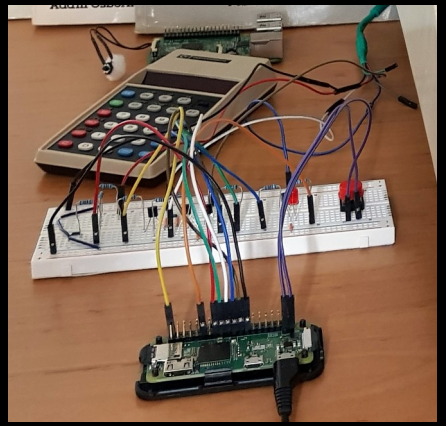


Small enhancements:

- remove 8+3 limitation
- !$ without overwrite
- let !$ show address & size of PRGs
- increase loading speed of wedge PRGs (by making them smaller)
- show some response on Commodore for some commands

THE FUTURE





Small enhancements:

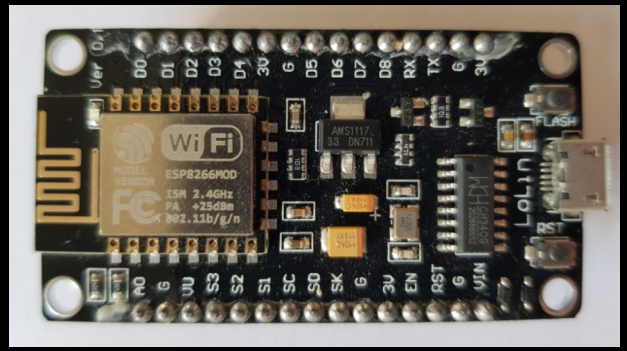
- remove 8+3 limitation
- !$ without overwrite
- let !$ show address & size of PRGs
- increase loading speed of wedge PRGs (by making them smaller)
- show some response on Commodore for some commands
- …

THE FUTURE

Ports:

THE FUTURE

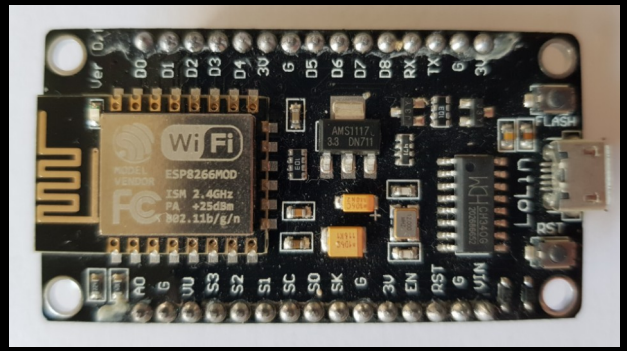
Ports: ESP8266



THE FUTURE

Ports: ESP8266

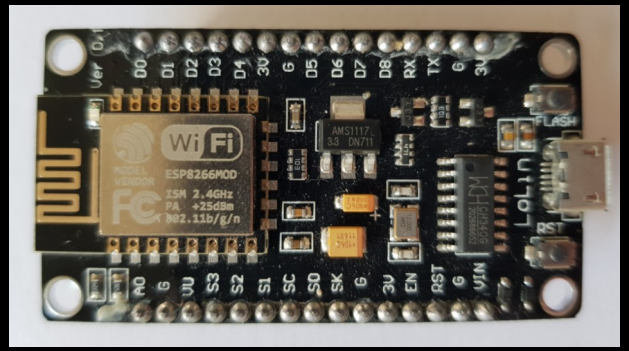
– using Arduino via PlatformIO



THE FUTURE

Ports: ESP8266

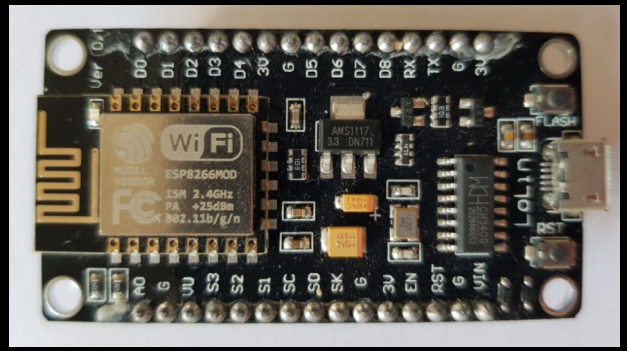
- using Arduino via PlatformIO
- next step: Compatibility mode LOAD



THE FUTURE

Ports: ESP8266

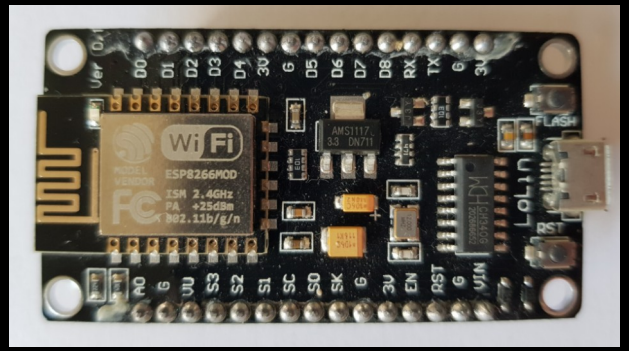
- using Arduino via PlatformIO
- next step: Compatibility mode LOAD
- SD card "reader" must be built

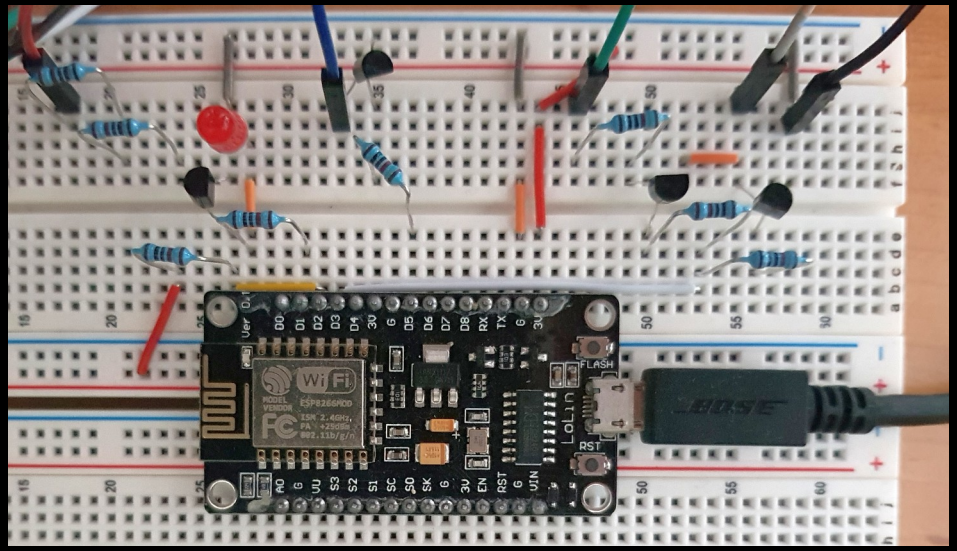


THE FUTURE

Ports: ESP8266

- using Arduino via PlatformIO
- next step: Compatibility mode LOAD
- SD card "reader" must be built

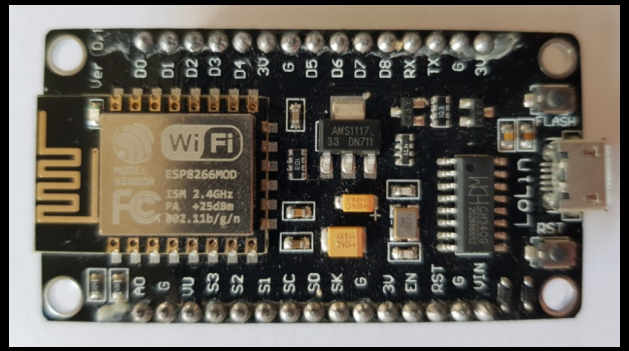


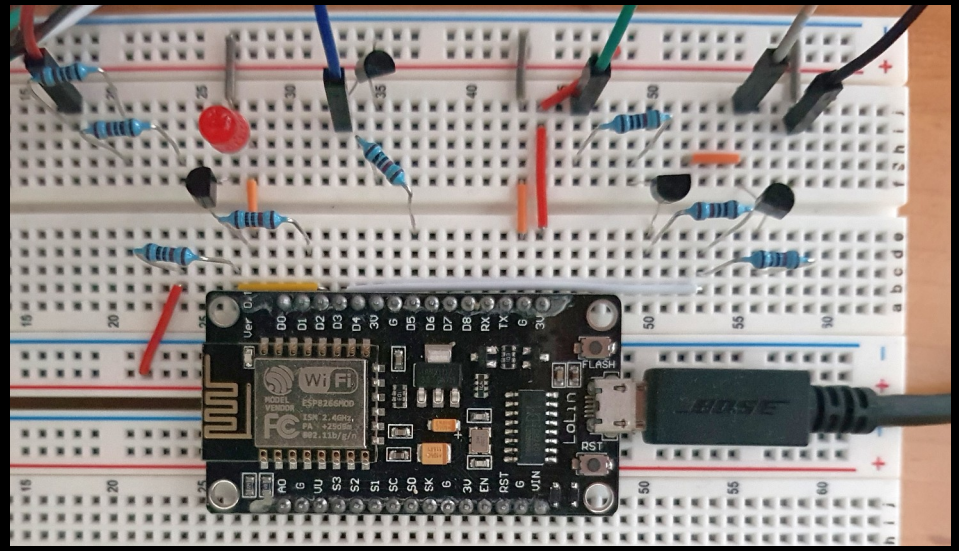


THE FUTURE

Ports: ESP8266

- using Arduino via PlatformIO
- next step: Compatibility mode LOAD
- SD card "reader" must be built



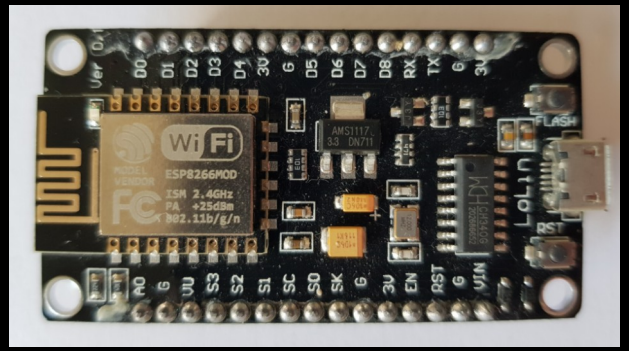


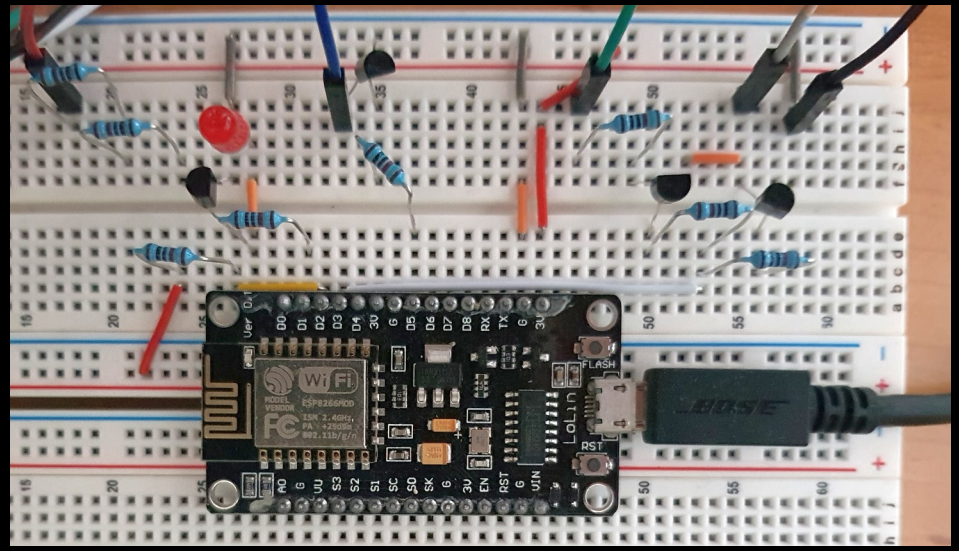
=> Super simple WiFi!

THE FUTURE

Ports: ESP8266

- using Arduino via PlatformIO
- next step: Compatibility mode LOAD
- SD card "reader" must be built





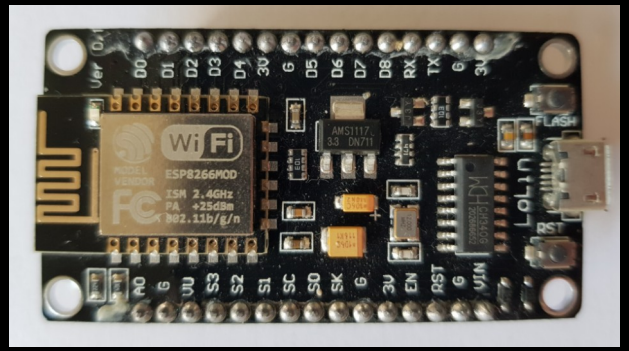
=> Super simple WiFi!

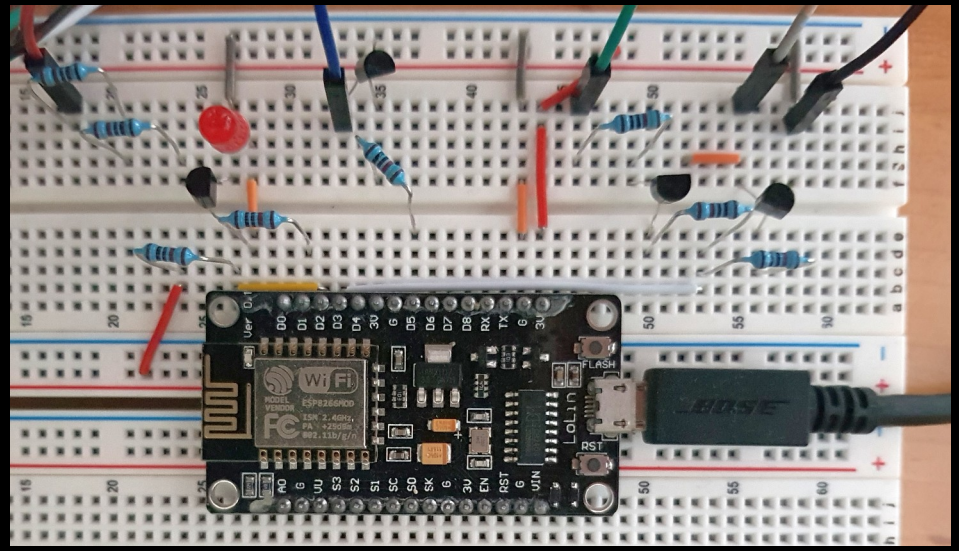
=>=> Internet!

THE FUTURE

Ports: ESP8266

- using Arduino via PlatformIO
- next step: Compatibility mode LOAD
- SD card "reader" must be built





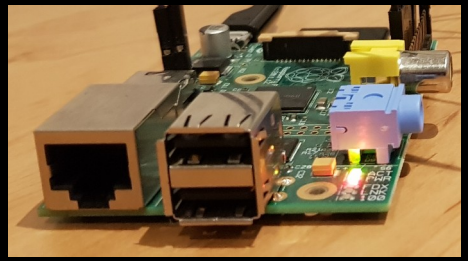
=> Super simple WiFi!

=>=> Internet!

=>=>=> World Domination!

THE FUTURE

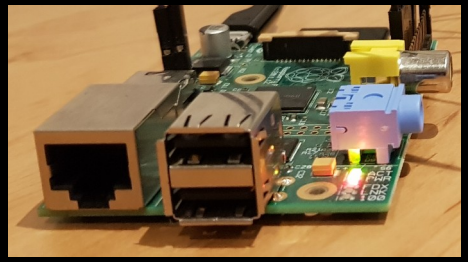
Ports: Raspbian/Linux



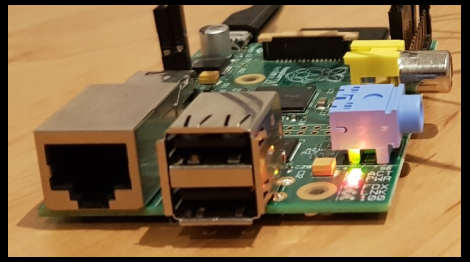
THE FUTURE

Ports: Raspbian/Linux

- problem: Scheduler vs. compatibility mode



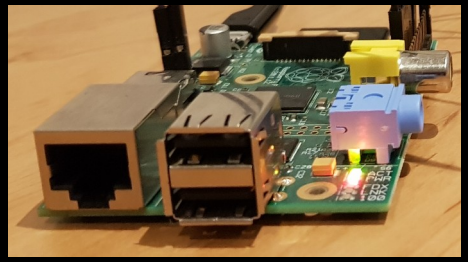
THE FUTURE



Ports: Raspbian/Linux

- problem: Scheduler vs. compatibility mode
- initial wedge transfer needs comp. mode

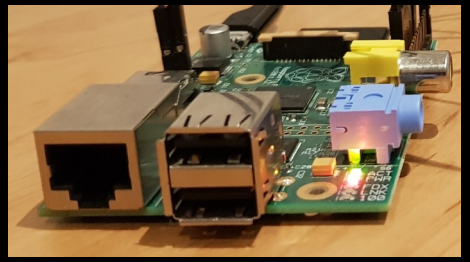
THE FUTURE



Ports: Raspbian/Linux

- problem: Scheduler vs. compatibility mode
- initial wedge transfer needs comp. mode
- solution: DMA!

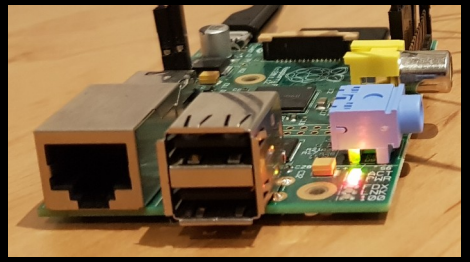
THE FUTURE



Ports: Raspbian/Linux

- problem: Scheduler vs. compatibility mode
- initial wedge transfer needs comp. mode
- solution: DMA!
- first LOAD test successful (using pigpio)

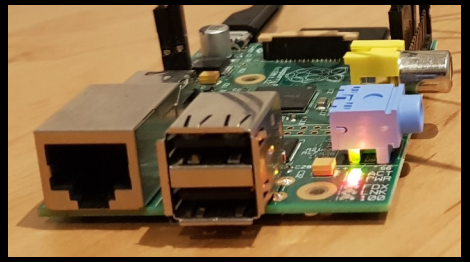
THE FUTURE



Ports: Raspbian/Linux

- problem: Scheduler vs. compatibility mode
- initial wedge transfer needs comp. mode
- solution: DMA!
- first LOAD test successful (using pigpio)
- own implementation necessary (no pigpio)

THE FUTURE

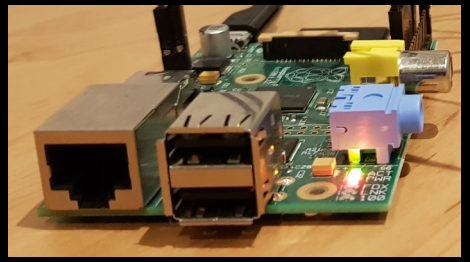


Ports: Raspbian/Linux

- problem: Scheduler vs. compatibility mode
- initial wedge transfer needs comp. mode
- solution: DMA!
- first LOAD test successful (using pigpio)
- own implementation necessary (no pigpio)
- next step:

  First LOAD test with own DMA implementation

THE FUTURE

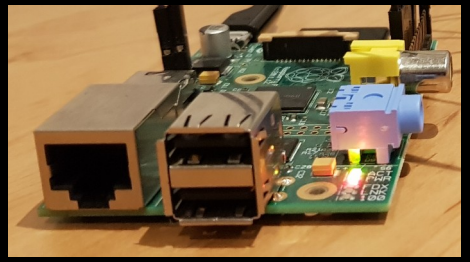


Ports: Raspbian/Linux

- problem: Scheduler vs. compatibility mode
- initial wedge transfer needs comp. mode
- solution: DMA!
- first LOAD test successful (using pigpio)
- own implementation necessary (no pigpio) => WiFi!
- next step:

  First LOAD test with own DMA implementation

THE FUTURE

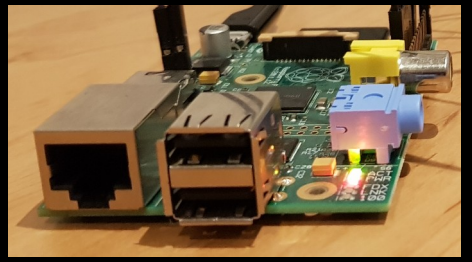


Ports: Raspbian/Linux

- problem: Scheduler vs. compatibility mode
- initial wedge transfer needs comp. mode
- solution: DMA!
- first LOAD test successful (using pigpio)
- own implementation necessary (no pigpio) => WiFi!
  => Internet!
- next step:

  First LOAD test with own DMA implementation

THE FUTURE



Ports: Raspbian/Linux

- problem: Scheduler vs. compatibility mode
- initial wedge transfer needs comp. mode
- solution: DMA!
- first LOAD test successful (using pigpio)
- own implementation necessary (no pigpio) => WiFi!
- next step: => => Internet!

  First LOAD test with own DMA implementation => => => World Domination (again..)!

SOURCE CODE & DOCUMENTATION

github.com/RhinoDevel/cbmtapepi

SOURCE CODE & DOCUMENTATION

*** THANKS FOR WATCHING ***

github.com/RhinoDevel/cbmtapepi