Table of Contents

1. Laboratory on the Internet of Things - Advanced IoT Hardware 2
1.1. Hardware 2
1.1.1. ESP32 WROVER DEVKIT 2
1.1.1.1. Built-in integrated LCD (TFT) display 3
1.1.1.2. Conûguring VREL - platformio.ini 4
1.1.2. TTGO T5 v.2.2 e-Paper 5
1.1.2.1. Built-in integrated e-Paper display 6
1.1.2.2. Conûguring VREL - platformio.ini 6
1.1.3. A special notes on PWM in ESP32 8
1.2. Laboratory scenarios 9
1.2.1. Task 1 9
1.2.2. Task 2 10
1.2.3. Task 3 10
1.2.4. Task 4 10
1.2.5. Report 10
1.3. Resources 10

1.Laboratory on the Internet of Thi Advanced IoT Hardware

Laboratory guide for IoT AIH students on ESP32 programming, using V of ESP32 modules:

- ESP32 WROVER DEVKIT module - that includes (among others) dot
- TTGO T5 v 2.2 module - that contains a 2.9 Waveshare e-paper, tw

You can access the distant laboratory (VREL) at http://distantlab.eu/
You need to register an account (unless you have one, already). A sho creation process and first steps at https://youtu.be/0R6oUWkBrpQ

1.1 Hardware

You will use laboratory nodes #25 and #20:

- TTGO with e-Paper module is VREL node #25
- WROVER DEVKIT with LCD TFT module is VREL node #20

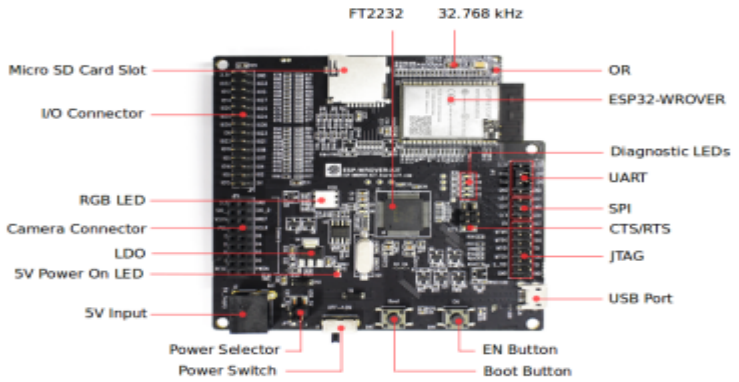
Below we present hardware details on the two modules:

1.1.1 ESP32 WROVER DEVKIT

This board is double-sided; on one side, you will find an LCD screen (back) and on the other side (front) there are GPIOs connectors you connect sensors and actuators t SPI reader and an RGB LED (among others).
Please note you can directly access only the module via the VREL node boards provided by the

Fig. 1 ESP32 WROVER Dev Kit fro [1)]

Fig. 2 ESP32 WROVER Dev Kit bac[2]



ESP32 provides a number of GPIOs to the developers, but some of the features as connected on the development board.

Some of the GPIOs are dedicated to implementing a limited n some are dedicated to using a particular protocol on this de development board contains some peripherals (RGB LED, LCD use GPIOs that are controlling them.

The following list presents GPIOs used with the integrated hardware:

Tab. 1 ESP32 WROVER DEVKIT GPIO and features

No	Feature	ESP32 GPIO Pin	Notes
1	JTAG	GPIO12-15	
2	Camera	GPIO0, GPIO4-5, GPIO18-19, GPIO21-23, GPIO25-27, GPIO34-36, GPIO39	If you do not use the ca Free RTOS, to use those G unless they're not share with another feature.
3	RGB LED	GPIO0, GPIO2, GPIO4	
4	MicroSD card reader	GPIO2, GPIO4, GPIO12-15, GPIO21	
5	LCD	GPIO5, GPIO18-19, GPIO21-23, GPIO25	

Note - GPIOs 34, 35, 36 and 39 do not have internal pull-up no

1.1.1. Built-in integrated LCD (TFT) display

The built-in display is ILI9341 TFT display, 320×240 pixels (240×320 a Although it is possible to program it raw, we will use a ready software display uses 16-bit per pixel memory organisation, frequently referenc bits, B-5-bits).

1.1.1.2. Conûguring VREL - platformio.ini

Correct conûguration is necessary to build and deploy your code.
For ESP32 WROVER DEVKIT, a good starting point is the following platformio.ini conûle:

```
[env:d1_mini]
platform = espressif32
board = esp-wrover-kit
framework = arduino
lib_ldf_mode = deep+
lib_deps =
   adafruit/Adafruit GFX Library
   Wire
   espressif/WROVER KIT LCD @ ^1.0.3
```

1.1.1.2. Practice: handling LCD display

Write Hello World on the LCD screen.
Using an LCD and a camera, you can observe it through, are your only
Therefore, please get familiar with handling LCD screens and give it a
below.

Type textual Hello World on the screen. Then, you're welcome to dec
orientation, etc.

To use LCD, you must include libraries. The beginning of your code should the
more-less following way:

```
...
#include <Arduino.h>
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <WROVER_KIT_LCD.h>
...
```

Then deûne a controller class that manages your LCD:

```
...
WROVER_KIT_LCD tft;
...
```

Remember to initiate the LCD controller class:

```
...
  tft.begin();
...
```

Some useful functions to handle LCD operations are presented below (

```
...
  tft.fillScreen(TFT_BLACK); //Clear screen
  tft.setCursor(0, 0); //Set upper left corner
  tft.setTextColor(TFT_WHITE); //Set text colour
  tft.setTextSize(2); //Set font size
  tft.println("Hello World!"); // Print the text
...
```

1.1.2. TTGO T5 v.2.2 e-Paper

TTGO (aka LILYGO) ESP32 module with integrated e-Paper 2.9-inch di
Paper development. Here you will learn the basics of handling the e-P
coloured (black/white) with 296×128 pixels resolution. The display is
bus
The TTGO T5 module also contains a speaker (buzzer), SPI TF card re
advertised as an electronic badge.



Fig. 3: TTGO T5 v2.2 front side with e-Paper display



Fig. 1: TTGO T5 v2.2 back side

ESP32 provides a number of GPIOs to the developers, but some of the features as connected on the development board.

> Some of the GPIOs are dedicated to implementing a limited n some are dedicated to using a particular protocol on this development board contains some peripherals (RGB LED, LCD not use GPIOs that are controlling them.

Tab. 2: TTGO T5 v2.2 GPIO and features

No	Feature	ESP32 GPIO Pin	Notes
1	e-Paper SPI interface	GPIO4, GPIO12, GPIO19, GPIO5, GPIO23, GPIO18	In order: ePaper BUSY, ePaper R DC, EPaper CS, EPaper MOSI, EP
2	MicroSD card reader	GPIO13, GPIO23, GPIO2, GPIO18	In order: SDCard CS, SDCard MO MISO, SDCard SCLK
3	Buttons	GPIO37, GPIO38, GPIO39	In order buttons 1,2,3
4	Speaker	GPIO25	Channel 0

1.1.2. Built-in integrated e-Paper display

The e-Paper (e-Ink) display is a bi-coloured, 296×128 pixels with an S manufactured by Waveshare. It has an integrated SPI controller[3]. There are various software libraries able to handle e-Paper displays. the moment is the GxEPD library that includes device drivers for the m displays. The exact model (as to be included in the code is Github GxEPD. Library https://github.com/lewisxhe/GxEPD/tree/master/src/GxGDEH029A1

1.1.2.2. Conûguring VREL - platformio.ini

Correct conûguration is necessary to build and deploy your code. For ESP32 TTGO T5 v2.2, a good starting point is the following conûg that is the platformio.ini

```
[env:d1_mini]
platform = espressif32
board = esp-wrover-kit
framework = arduino
lib_deps = adafruit/Adafruit-GFX-Library
    Wire
    zinggjm/GxEPD
```

1.1.2.2. Practice: handling e-Paper display

Write Hello World on the e-Paper screen.
Using an e-paper display and a camera, you can observe it through, ar
TTGO. Therefore, please get familiar with handling e-Paper and give i
below.

Type textual Hello World on the screen. Then, you're welcome to dec
orientation, experiment with bitmap displaying, partial updates and so

To use e-Paper, you must include libraries in the beginning of the source code. The code should t
look more-less following way:

```
...
#include <Arduino.h>
#include <GxEPD.h>
#include <Adafruit_I2CDevice.h>
#include "SPI.h"
#include <GxIO/GxIO_SPI/GxIO_SPI.h>
#include <GxIO/GxIO.h>

//TTGO e-Ink T5 EP32 v 2.2. board specific:
#include <GxGDEH029A1/GxGDEH029A1.h> // 2.9" b/w
#define ELINK_BUSY 4
#define ELINK_RESET 12
#define ELINK_DC 19
#define ELINK_SS 5

#define SPI_MOSI 23
#define SPI_MISO 2
#define SPI_CLK 18
...
```

Note, by the e-Paper related display drivers; there is also a font deû n
included:

```
...
#include <Fonts/FreeSans9pt7b.h>
...
```

and then deû ne this font as a default one to use code below:

```
...
#define DEFALUT_FONT FreeSans9pt7b
...
```

Then deû ne a controller class that manages your e-Ink display:

```
...
GxIO_Class io(SPI, ELINK_SS, ELINK_DC, ELINK_RESET);
GxEPD_Class display(io, ELINK_RESET, ELINK_BUSY);
...
```

Some useful functions to handle e-Paper operations are presented bel

```
...
  SPI.begin(SPI_CLK, SPI_MISO, SPI_MOSI, -1);
  display.init();
  display.setRotation(1);
  display.eraseDisplay();
  display.setTextColor(GxEPD_BLACK);
  display.setFont(&DEFALUT_FONT);
  display.setTextSize(0);

  display.fillScreen(GxEPD_WHITE);
  display.setCursor(20, 20);
  display.println("Hello world!");
  display.update();
...
```

Note, for ease, predeû ned values present black (`GxEPD_BLACK`) and white (`GxEPD_WHITE`) colour displ

> You need to use a font to draw on the e-Ink screen. Use one o along with the GxEPD library.

> In the laboratory node #25, the e-ink display has been mount to keep text horizontal, you need to use:
>
> ```
> display.setRotation(1);
> ```

1.1.3 A special notes on PWM in ESP32

PWM is frequently used to control analogue-style, eû cient voltage on t resistance driver, PWM uses pulses to change adequate power, deliver motors, but also to LEDs, bulbs, heaters, and indirectly to the servos way, see lectures).

The classical `analogWrite` method known as Arduino (Uno, Mega) and ESP8266
ESP32 has sixteen (0 to 15) PWM channels (controllers) that can be fr
GPIOs. Moreover,
ESP32 can use various resolutions of the PWM: from 8 bit (0..255) to
Arduino uses only 8 bit one.

To use PWM in ESP32, one must perform the following steps:

- Conû gure GPIO `OUTPUT`
- Initiate PWM controller with û xing PWM frequency and resolution
- Bind controller to the GPIO pin
- Write to the controller (not to the PIN!) providing duty cycle relate above - every call persistently sets PWM duty cycle until there is setting duty cycle.

1.2 Laboratory scenarios

In general, you will program using the Arduino framework, remotely via
VREL system. Note that your only output uses the web camera, so resu
way.
As presented during the lecture, you will also use FreeRTOS functions
programming to perform tasks, as demonstrated during the lecture.

> ESP32 framework uses C++ optimiser while compiling the sou
> that C++ traces unused code and removes/optimises it, to
> your tasks to be completed, involve integer and û oating-point
> it is essential to use the result somewhere, i.e. do a dummy
> on the screen. Otherwise, the C++ compiler may remove this s
> you won't be able to obtain reasonable results, because yo
> trunk and not executed at all!

> FreeRTOS, once the task is created and scheduled for exec
> inû nitely! It means, your function implementing the task wil
> again. If you intended to run a looped task, that is pretty OK.
> one-oû task, then you need explicitly call:
>
> ```
> ...
> vTaskDelete(NULL);
> ...
> ```
>
> as a last line of your function implementing the task.

1.2.1 Task 1

Evaluate single-core performance for integer computation. Perform two

to the APP core and separately to the PRO core. First, observe if ther measurements. Then, propose an algorithm that can generate a comple observable and measurable. Perform at least 10 measurements for eac parts of code for the Dhrystone benchmark: https://en.wikipedia.org/wiki/Dhrystone.

1.2.2 Task 2

Evaluate multicore performance for integer computation. Perform singl least 4 times (any even number works here) your algorithm. Then exec if there is a difference in measurements. Propose an algorithm that ca integer computation observable and measurable. Perform at least 20 m experiment (10 serial + 10 parallel). Consider using parts of code for https://en.wikipedia.org/wiki/Dhrystone

1.2.3 Task 3

Evaluate single-core performance for floating-point computation. Perfo bound to the APP core and separately to the PRO core. First, observe measurements. Then, propose an algorithm that can generate a comple computation that is observable and measurable. Perform at least 10 m experiment. Consider using FP parts of code for Whetstone benchmark https://en.wikipedia.org/wiki/Whetstone_(benchmark)

1.2.4 Task 4

Evaluate multicore performance for floating-point computation. First, p computation, running your algorithm at least 4 times (any even numbe Then execute in parallel (i.e. 2×2). Finally, propose an algorithm that floating-point computation that is observable and measurable. Perform each experiment (10 serial + 10 parallel). Consider using FP parts of https://en.wikipedia.org/wiki/Whetstone_(benchmark)

1.2.5 Report

Include algorithm definition and measurements, along with statistical a average and standard deviation) and a summary.

1.3 Resources

1. ESP32 WROVER kit starting page
2. Hardware definition (pinouts, gpios, etc.)
3. ESP-WROVER-KIT V4.1 schematic
4. https://github.com/LilyGO/TTGO-Electronic-Badge
5. https://github.com/lewisxhe/TTGO-EPaper-Series

6. https://www.waveshare.com/wiki/2.9inch_e-Paper_Module
7. https://github.com/lewisxhe/GxEPD
8. On-line bitmap (image) to memory content (cpp) converter
9. FreeRTOS - multitasking with ESP32 explained

1) 2)
,
https://docs.espressif.com/projects/esp-idf/en/latest/hw-reference/get
3)
https://www.waveshare.com/wiki/2.9inch_e-Paper_Module

From:
https://dokuwiki.zmitac.aei.polsl.pl/ - RAU6 Dokuwiki

Permanent link:
https://dokuwiki.zmitac.aei.polsl.pl/en:edu:zmitac:s2:iot:163733

Last update: 2021/11/19 15:32