**Assessment type (🗹):**

Questioning (Oral/Written)

Practical Demonstration

3rd Party Report

Other – Project/Portfolio (*please specify)*

Practical Portfolio work with embedded knowledge questions

**Assessment Resources:**

|  |
| --- |
| The base requirements this assessment task include:   * Web server, Python interpreter and database server * IDE or editor for developing Python programs (only PyCharm supported by the college) * Raspberry Pi with SenseHat or other IoT devices, like Arduino Uno or ESP32 * Access to Office 365 & Microsoft Word * Report Template (Portfolio: AT2 POR-Task-3 Template) as supplied   Use of some of these items may not occur in this part of the assessment task. |

**Assessment Instructions:**

|  |
| --- |
| Follow the steps listed in this assessment item.  Submission of the documentation, code, and associated items is at the end of each part of the portfolio.  Each part of the portfolio has a deadline for submission.  It is advantageous to you to attempt to meet the deadline provided.  **Scenario**  You are currently working for a small Perth-based start-up company called Incredibly Obvious Technologies. They are looking to create a presence in the home automation and monitoring market.  Download and read the complete “Portfolio-Task-Scenario” from Blackboard.  **General Instructions**  Use this document for your answers to this portfolio.  Rename the document as XXX-POR-Task-3.docx, replacing the XXX with your initials. For example, Adrian Gould would use AG-POR-Task-3.docx for his submitted filename.  **Answering Questions**  When a step includes a question, you must attempt to answer it.  There is a minimum and maximum number of words to use for each answer.  If a step has more than one question, these maxima and minima are a total for all the questions in that specific step.  All answers must be in complete sentences unless indicated.  If required, make sure to add any code you’ve written in a separate file to your submission. DO NOT put code in a Word document.  **Sources of Information**  In industry, it is good practice to keep track of where information was obtained. This is especially true if it is a written document, or even code.  If you answer any questions using information from web sites, please include the site name and URL (Web site address) after the answer. Likewise, include the title and author for books and magazine articles. For example:  • RS Electronics Ltd: <https://au.rs-online.com/>  • Slack API Documentation, Users List Method: <https://api.slack.com/methods/users.list>  **Code Storage**  We advise that you create a GIT repository on GitHub and use this to store a copy of your work. You may also use OneDrive within your college Office365 to store a backup of your code or keep a copy on a USB thumb drive. |

**Assessment Instrument:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | Step | Task to Perform | | --- | --- | | 00 | Prepare for Work Make sure you have followed the instructions on creating the answer document, as given in the General Instructions.  Familiarise yourself with the content and document your progress in this assessment.  Make sure that you complete the title page of the document.  At any stage during this assignment, you may consult the stakeholder(s) or their representative(s).  Please note that this assessment should be done on actual ESP32 hardware, which will be provided.  You can access the ESP32 API Reference here: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html>  This link specifically points to GPIO & RTC GPIO, but it will give you easy access to all the other areas of the API too. | |  | *This space left intentionally blank.* | |  | *This space left intentionally blank.* | | 01 | LEDs and Switches: start simple For this task, you will be provided (in class) with a simple electronic circuit built around an ESP32 microcontroller.  The schematic for it looks like this (simplified):    It contains the following elements:   * **U1**: ESP32 microcontroller * **R1**, **R2**: resistors * **D1**, **D2**: LEDs (one red – GPIO19, one green – GPIO18) * **SW1**: switch (reed relay – GPIO15)   The microcontroller is powered with 3.3V, which may come from a battery or from a USB connection.  Questions on following page. | | A01a | Q: What should the value of R1 and R2 be?   Referring to ESP32 tutorial, maximum current of Red LED is 20mA and of Green LED is 10mA. So, supplied voltage is 3.3V, one Red LED, one Green LED, and parallel connection.  R1(Red) = V/I = 3.3V / 0.02A = 165Ω  R2(Green) = V/I = 3.3V / 0.01A = 330Ω | | A01b | Q: How should each of the three GPIO pins be configured? GPIO 19 connected to the Red led as a positive output, GPIO 18 to the Green led, and GPIO 15 to the button switch reaching all to the Ground. | | A01c | Q: What additional requirements does GPIO15 have for it to work properly? When using GPIO15 pin, we need to ensure that any external pull-down resistor connected to GPIO15 is not interfering with our desired functionality. So, in the case, GPIO15 is configured as an input for the button switch, it needs to be held at a defined voltage level when the button is not pressed to prevent it from floating and causing unpredictable behaviour. | | 02 | Investigate the program Download the provided software for this task from Blackboard.  Investigate the program to get an idea of its workings.  Answer the questions below. | | A02a | Q: In your own words, describe what the program as a whole does. It seems this program is designed to interact with GPIO pins to control LEDs and read input from a door switch, checking the state of the door switch and indicating a visual state whether the door is open or closed. | | A02b | Q: What are the four main parts of the program? (Hint: look closely at the structure of the code.)  The four parts are:   * Inclusion * Definitions * Setup Function * Loop Function | | A02c | Q: What is the purpose of the function setup()? The setup() function is usually for initializing the program’s environment and configuring the hardware settings. In this program, the function is for serial communication initialization and GPIO configuration. | | A02d | Q: What is the purpose of the function loop()? The loop() function is to continuously execute a series of instructions or tasks as long as the ESP32 board is powered on. In this program the loop function is for leading input of the door switch state by using ‘digitalRead() and also for updating the LEDs to indicate the status of the door. Plus, it gives execution for 5 seconds of delays as well. | | 03 | Run the program You will be provided with a way to program (or “flash”) the ESP32 in the circuit.  To run the program on the ESP32, it must first be transferred into its memory.  Use the Arduino IDE to program the ESP32.  You will use the code from Question 2.  After flashing the program, you can experiment with the circuit to see what it does.  Provide screenshots of the process of compiling and uploading the code.  Take a short video of the circuit in action (no more than 15 seconds).  Ensure you speak your name clearly when videoing the circuit.  Save this video as XXX-AT2-Pt3-03.mp4 (mp4, vog, avi or similar allowed). Replace XXX with your initials. | | A03a | Screenshot(s) of Program in Action: | | A03b | Q: In your own words, describe what you observed when you experimented with the circuit. The program is designed to interact with door switch as an input and LEDs as output indicating the state of the door with different colour of LEDs. | | A03c | Q: What could be a major disadvantage of the program as it is? Potential disadvantage could be its blocking behaviour during the delay period in the loop function which call pauses the execution of any other tasks. Also, | | 04a | ZZZzzz (Yawn) We want to hook up our little device into a network and we want it to run off a battery.  But before we write the code that will connect the device (in some way) to the internet, we will make it a bit less power-hungry.  For this purpose, we will change the code so the device will go into “deep sleep” for as much of the time as it possibly can.  We will do this in a few steps.  The first step is to limit the power consumption of the LEDs.  We will do this by switching the LEDs on for a very brief period only, e.g., 50ms.  We can do this with only very minor changes to the code. | | 04b | ZZZzzz (Yawn) - continued Add the following function to your program:  void flash\_led(const int pin\_state) {  // Your code goes here  }  From loop(), move the two lines that turn on/off the LEDs based on pin\_state into the new function and replace them with the following line:  flash\_led(pin\_state);  Add a 50ms delay and place it in a suitable location in the new function. | | 04b | ZZZzzz (Yawn) - continued Revise how the LEDs are turned on/off so that only the appropriate LED is briefly lit: **green** when the door is **closed**, **red** when it’s **open**.  Talk your solution through with your lecturer.  Compile (but don’t yet flash) your program to see whether you have made any syntax errors. If so, correct those errors.  If there are no errors, flash your program and see if it works properly.  Provide screenshots of the process and answer the questions below.  Add this version of the program to your submission as main-v4.ino. | | A04a | Screenshot(s) of Program Compilation and Flash Process: | | A04b | Q: Considering only the power consumption of the LED, how much have you reduced it? (Provide a percentage.) Existing LED setting was 5000 milliseconds on but, now LED lights on just 50 milliseconds which means just only 1/100 compared to existing LED power consumption.  So now Green LED has 99% of reduction in power consumption. | | 05a | I need more sleep… Reducing the power consumption of the LEDs is a good first step. But our program is still running an infinite loop to do only one thing: check the status of the switch and flash an LED accordingly.  Not very efficient…  We will fix this by putting the device into a sleep mode.  As you know by now, device can be put in different types of sleep: “normal” sleep, deep sleep, and hibernate (and possibly others). Not all modes of sleep are supported by all devices.  For our solution, we will use the “deep sleep” functionality of the ESP32. This powers down many subsystems that we don’t need. We will wake the microcontroller if any of the following two situations occurs:   1. The input pin (connected to the switch) changes its state. 2. A predefined timeout occurs. Think of this is a health monitor, where we expect at least one update every N hours. This will be relevant when we connect the device to a network. | | 05b | I need more sleep… - continued Follow these instructions carefully:  Include the file driver/rtc\_io.h in the relevant section in your code. We need this because deep sleep involves the RTC (real-time clock) of the ESP32.  Make the following change in your code… first remove:  gpio\_set\_direction(GPIO\_SWITCH, GPIO\_MODE\_INPUT);  gpio\_set\_pull\_mode(GPIO\_SWITCH, GPIO\_PULLUP\_ONLY);  and replace these two lines with:  rtc\_gpio\_deinit**(GPIO\_SWITCH);**  This will reconfigure the GPIO pin for the switch as a GPIO pin after waking up.  Move the code from the function loop() into setup() (at the end) and remove the delay. Instead of waiting, we’ll go to sleep now. | | 05c | I need more sleep… - continued Add the following lines to your code, at the end of setup():  /\* Make sure to pull up the sensor input. \*/  rtc\_gpio\_pullup\_en(GPIO\_SWITCH);  /\* Set up the conditions for waking up. \*/  esp\_sleep\_enable\_ext0\_wakeup(GPIO\_SWITCH, pin\_state ^ 1);  Serial.println("ZZZzzz");  esp\_deep\_sleep\_start();  (Now you know why we initialised the serial port earlier.)  Have a good look at the few lines above.  Try to understand what’s going on. | | 05d | I need more sleep… - continued Compile the program and see whether any errors pop up. You may want to show your solution to the lecturer first before continuing.  When the program compiles without errors and warnings, you may flash it into the ESP32 memory and run the program.  Add the code to your submission as main-v5.ino.  Answer the questions that follow below: | | A05a | Q: What is the purpose of, and what is the importance of pin\_state ^ 1? In your own words, describe what the purpose is of pin\_state ^ 1 in the code above?  pin\_state ^ 1 is used to set up a wake-up condition based on a change in the state of an input pin during deep sleep, providing more flexibility in handling and adaptability to external events. | | A05b | What are the obvious differences between solutions in step 4 and 5? In main-v4, the setup function initializes the GPIO pin connected to the door switch as an input with pull-up mode but, in main-v5, these lines of code are commented out, meaning the door switch initialization is skipped.  In main-v5, after initializing the GPIOs for LED outputs, the ‘rtc\_gpio\_deinit()’ function is called to disable the RTC GPIO associated with ‘GPIO\_SWITCH’. Additionally, in main-v5, the code sets up conditions for waking up from deep sleep using ‘esp\_sleep\_enable\_ext0\_wakeup()’ function. These configurations are absent in main-v4. In main-v5, the loop function is entirely commented out, indicating that there’s no active looping behaviour. | | 06 | Wait, what? Wait! Our current solution wakes up the program when the state of the input pin changes.  But didn’t we say we would also wake up after a predefined timeout?  Indeed, we did say that, so let’s add that to the program.  Add the following line just after the setting the wakeup condition for the GPIO pin:  esp\_sleep\_enable\_timer\_wakeup(SLEEP\_FOR\_US);  Add a #define with a useful value for SLEEP\_FOR\_US, e.g., the equivalent of 1 hours.  (For testing purposes, you could set it to a much shorter period, e.g., 20 seconds. Also, note that the timeout is in microseconds, so it will be a rather big number.)  Compile your program and if there are no errors and warnings, flash the ESP32.  Observe the behaviour, including the timer wakeup.  Add the code to your submission as main-v6.ino. | | A06 | No Answer required for step 6 | | 07a | Who woke me? Our solution now uses deep sleep to save as much power as possible. It also occasionally wakes up to update the status, which will be of interest when we connect the device to a network.  As cherry on the cake, we’re going to find out what the cause is of the microcontroller waking up.  We have provisions for two sources, the GPIO pin changing state, and the timeout.  The program could, however, be run when it comes out of a reset situation.  Add the following line to your code, just before reading the GPIO pin:  print\_wakeup\_cause();  Then add a new function with the same name. It should return void and not accept any arguments.  Within the new function, use a switch-case statement to check the result from a call to the function esp\_sleep\_get\_wakeup\_cause().  The expected return values are ESP\_SLEEP\_WAKEUP\_EXT0 and ESP\_SLEEP\_WAKEUP\_TIMER.  Print a suitable message for each of these wakeup sources using Serial.println().  Use the switch-case’s default option to print a message that the source was neither the GPIO pin nor the timer. | | 07b | Who woke me? - Continued Add screenshots as evidence that your program can determine the source of it waking up from deep sleep.  Take a short video of the circuit in action (no more than 30 seconds).  Ensure you speak your name clearly when videoing the circuit.  Save this video as XXX-AT2-Pt3-07.mp4 (mp4, vog, avi or similar allowed). Replace XXX with your initials.  Add a version of your code to your submission as main-v7.ino. | | 07c | Who woke me? - Continued Add to your program a line that prints the status of the switch to the serial port at the appropriate time. | | A07a | Screenshot(s) of Program Compilation and Flash Process: | | End | Submission of Portfolio Work To submit the portfolio, do the following:   * Save the document with your answers as a MS Word file (.docx). * Put all your code inside a zip-file. * Open Blackboard, and locate the AT2 Portfolio Task 1 assessment * Open the assessment and upload the original word-processed document. * Click submit.   Whilst there is no need to use any other word processing software as you have access to Office 365 for free as a student, if you use Apple Pages, or Open Office, we will then require you to upload the original file AND a PDF version. | |

# Appendix A: Code Style Guidelines

The following guidelines should be applied to your code as it is developed.

Many may be applied via the use of PhpStorm, PyCharm or similar plugins and code formatting.

### PHP Code (General)

Please refer to the PHP PSRs:

* <https://www.php-fig.org/psr/psr-1/>
* <https://www.php-fig.org/psr/psr-12/>
* <https://www.php-fig.org/psr/psr-4/>

### Applications Built with Laravel

Please refer to the following articles:

* <https://dev.to/lathindu1/laravel-best-practice-coding-standards-part-01-304l>
* <https://dev.to/lathindu1/laravel-best-practice-coding-standards-part-02-a40>

### HTML Code

Please refer to the Google Style guide:

* <https://google.github.io/styleguide/htmlcssguide.html>

### JavaScript

Refer to the Google JS Style guide:

* <https://google.github.io/styleguide/jsguide.html>

### Python Code

Your code will follow the PEP 8 standard.

Other code standards available in the Presentation, “Python Coding Standards for North Metropolitan TAFE”.

### JSON Code

JSON should be formatted in an appropriate manner.