Microcontroller Block Validation

Block Champion: Joseph Conrow
Date: Mar-08-2024

1. Description

The microcontroller will be the main interconnected piece of the entire project. As it is the main communication device and interfaces between all the other functional blocks. It is crucial to determine the appropriate limitations and functionality that the specific model and module chosen has and can possibly develop. The overall final decision model we chose was the ESP32-WROOM–32 module, AKA "DOIT ESP32 DevKit V1 Wi-Fi Development Board". This way, with a pre-built chip design, the ESP32 chip itself can easily be accessed by our other components very easily using pin headers. The power supply pin also was able to regulate the specific power demands of our design. As the voltage regulator on the module allowed us to focus all of our power consumption from a single component source, as the method of supply to a standardized system voltage, that is, 5 Volts.

Some of the more specific rationale we used to pick this board were guided by their compatibility in communication protocol in reference to our selected project graphical display component, speaker component, internet component, and control components. These modes of communication involved SPI, I2C, PWM, WiFi, and DC signals. These are all protocols that the module chosen can provide with the ESP32 chip soldered on. The power draw of 3.3 volts is also something that the display is able to run off of, which the ESP32 can supply, allowing all of the communications and connections of the display to be rooted to the microcontroller. The connections to the web interface, and thus the user/clock/alarm server database is also to be accessed solely via the microcontroller. The speaker is also able to run and power distinctly sourced from the microcontroller, giving it the correct signals and internet connection to source a radio sound source if such audio source is chosen. The buttons also are a direct interworking interface for the microcontroller, as it can supply itself with a signal voltage and a receiving pin to recognize the state of such user buttons.

The inner workings of the microcontroller bring all of these components together to work in harmony. By receiving signals from the database, the display can be correctly displaying the correct time value. It conducts the appropriate sound values and timings for the alarm based on the stored website alarm settings. The microcontroller will be sustainable within the state of offline connection as well. One of the project objectives is to deliver the same functionality despite the source of power or internet connection. The small physical footprint of the chosen microcontroller is also desirable and easy to fit within our designed product enclosure. As this chosen microcontroller module is apt to function within each specific interface of the project components, the ability of it is also able to manage, store information, and send it where it needs to go, in a proper and useful manner, allowing the user to experience a clean, and seamless interaction with the product, regardless of which end component they are conducting interaction with, whether that be the graphical clock display, the website interface, the control buttons, or the audio of the speaker.

2. Design

The overall design and implementation steps are based on component specifications. This includes the power demands and communication protocols of each interface. The black box diagram simply refers to the interface setup for inputs and outputs. The ESP32-WROOM-32 Module Pinout explains the selected module's functionality, and interfacing capabilities, in reference to needs of each component. The Component Connection Pinout is meant to explain the physical connections between the microcontroller and its other components, proving capabilities to maintain connection between each, properly. The ESP32-WROOM-32 Chip DC Output figure is a brief excerpt from the microcontroller module in explanation for proof of the power demands for the ESP32 chip itself, which is comparable with the power regulation of the chip's accompanied module.

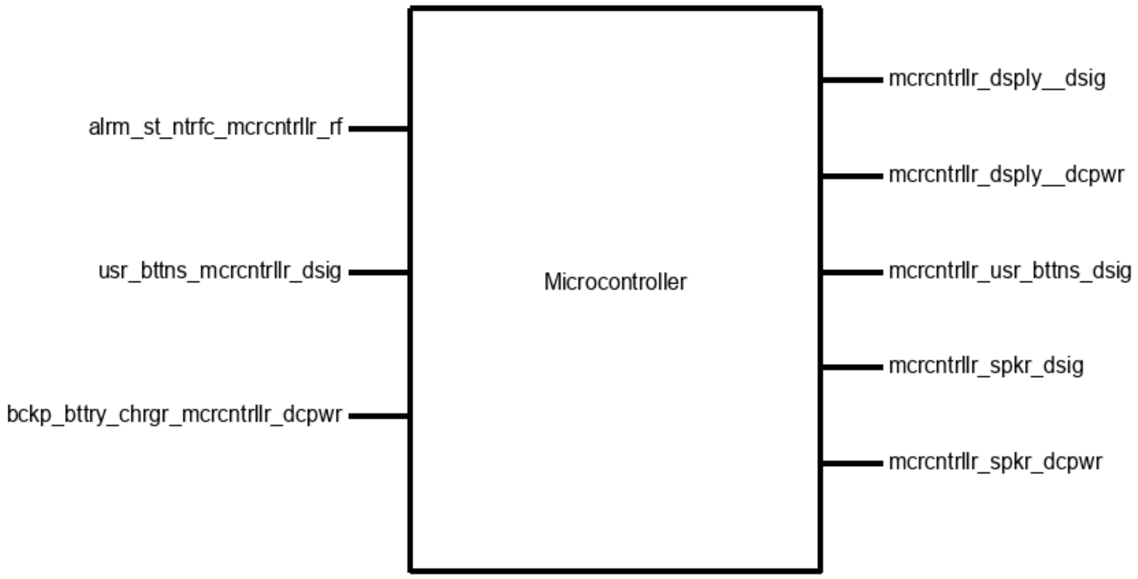
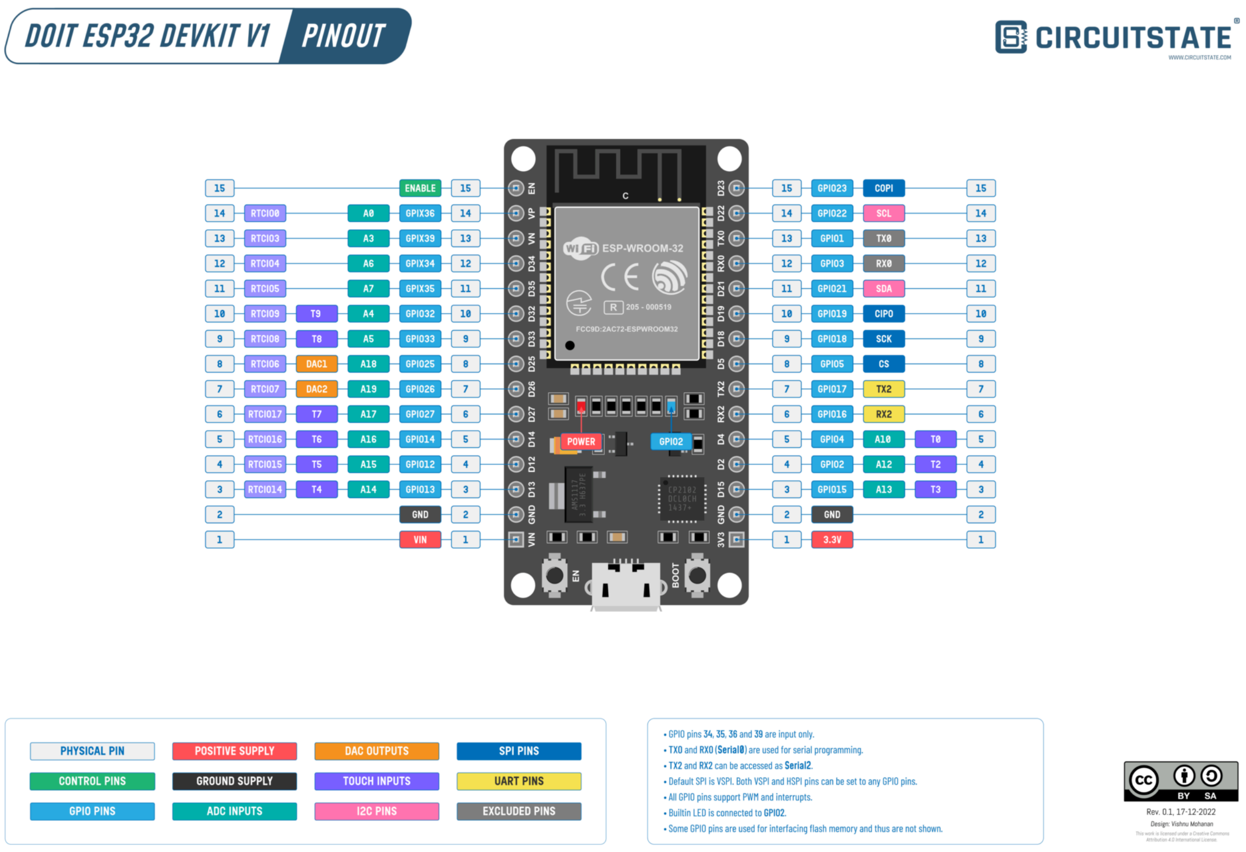

alrm_st_ntrfc_mcrcntrllr_rf
usr_bttns_mcrcntrllr_dsig
bckp_bttry_chrgr_mcrcntrllr_dcpwr
Microcontroller
mcrcntrllr_dsply__dsig
mcrcntrllr_dsply__dcpwr
mcrcntrllr_usr_bttns_dsig
mcrcntrllr_spkr_dsig
mcrcntrllr_spkr_dcpwr

Fig. X. ESP32 Black Box

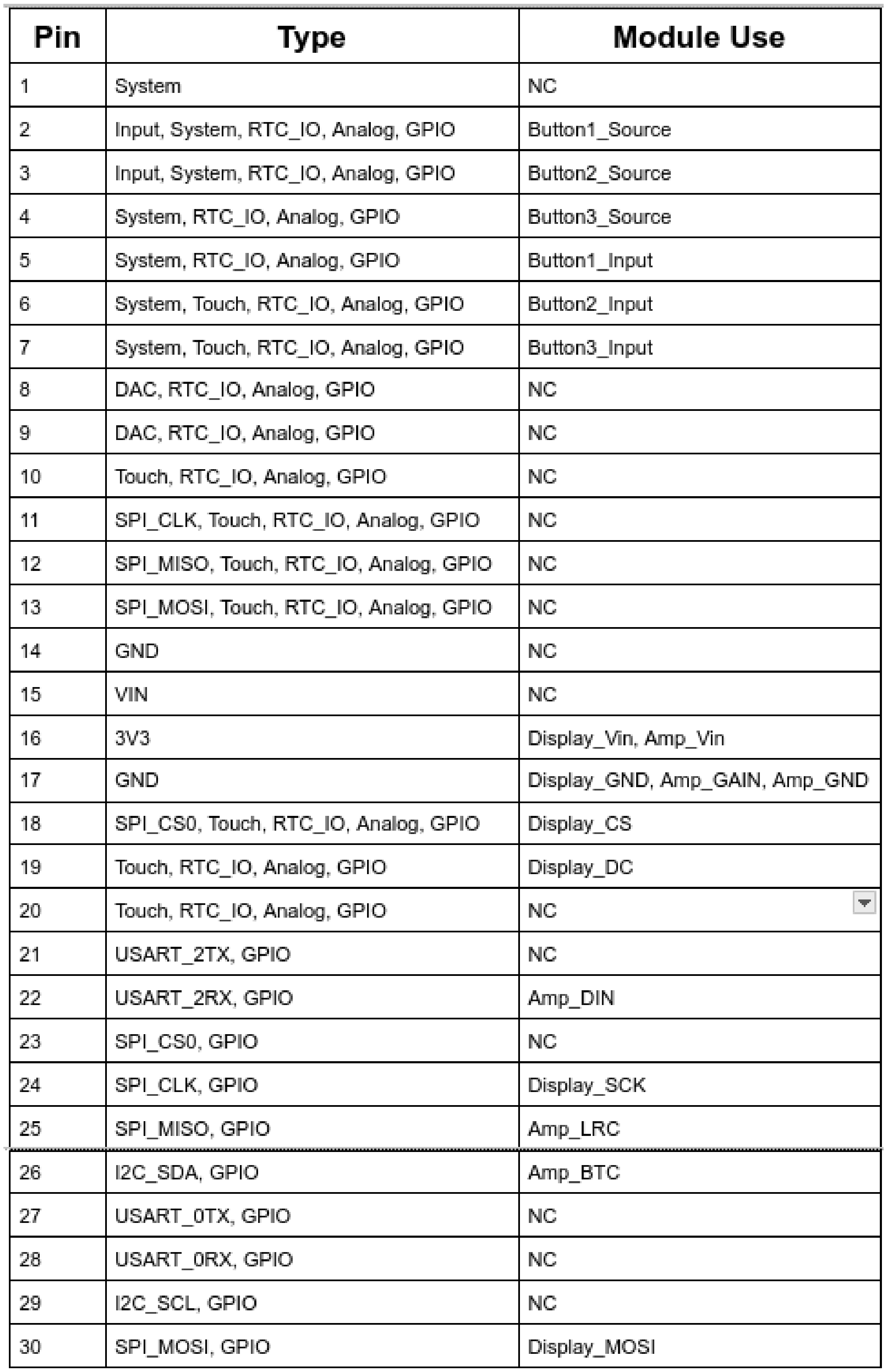


- GPIO pins **34**, **35**, **36** and **39** are input only.
- **TX0** and **RX0** (**Serial0**) are used for serial programming.
- **TX2** and **RX2** can be accessed as **Serial2**.
- Default SPI is VSPI. Both VSPI and HSPI pins can be set to any GPIO pins.
- All GPIO pins support PWM and interrupts.
- Builtin LED is connected to **GPIO2**.
- Some GPIO pins are used for interfacing flash memory and thus are not shown.

Rev. 0.1, 17-12-2022

Design: Vishnu Mohanan

Fig. X. ESP32-WROOM-32 Module Pinout [1]



Pin	Type	Module Use
1	System	NC
2	Input, System, RTC_IO, Analog, GPIO	Button1_Source
3	Input, System, RTC_IO, Analog, GPIO	Button2_Source
4	System, RTC_IO, Analog, GPIO	Button3_Source
5	System, RTC_IO, Analog, GPIO	Button1_Input
6	System, Touch, RTC_IO, Analog, GPIO	Button2_Input
7	System, Touch, RTC_IO, Analog, GPIO	Button3_Input
8	DAC, RTC_IO, Analog, GPIO	NC
9	DAC, RTC_IO, Analog, GPIO	NC
10	Touch, RTC_IO, Analog, GPIO	NC
11	SPI_CLK, Touch, RTC_IO, Analog, GPIO	NC
12	SPI_MISO, Touch, RTC_IO, Analog, GPIO	NC
13	SPI_MOSI, Touch, RTC_IO, Analog, GPIO	NC
14	GND	NC
15	VIN	NC
16	3V3	Display_Vin, Amp_Vin
17	GND	Display_GND, Amp_GAIN, Amp_GND
18	SPI_CS0, Touch, RTC_IO, Analog, GPIO	Display_CS
19	Touch, RTC_IO, Analog, GPIO	Display_DC
20	Touch, RTC_IO, Analog, GPIO	NC
21	USART_2TX, GPIO	NC
22	USART_2RX, GPIO	Amp_DIN
23	SPI_CS0, GPIO	NC
24	SPI_CLK, GPIO	Display_SCK
25	SPI_MISO, GPIO	Amp_LRC
26	I2C_SDA, GPIO	Amp_BTC
27	USART_0TX, GPIO	NC
28	USART_0RX, GPIO	NC
29	I2C_SCL, GPIO	NC
30	SPI_MOSI, GPIO	Display_MOSI

Fig. X. Component Connection Pinout

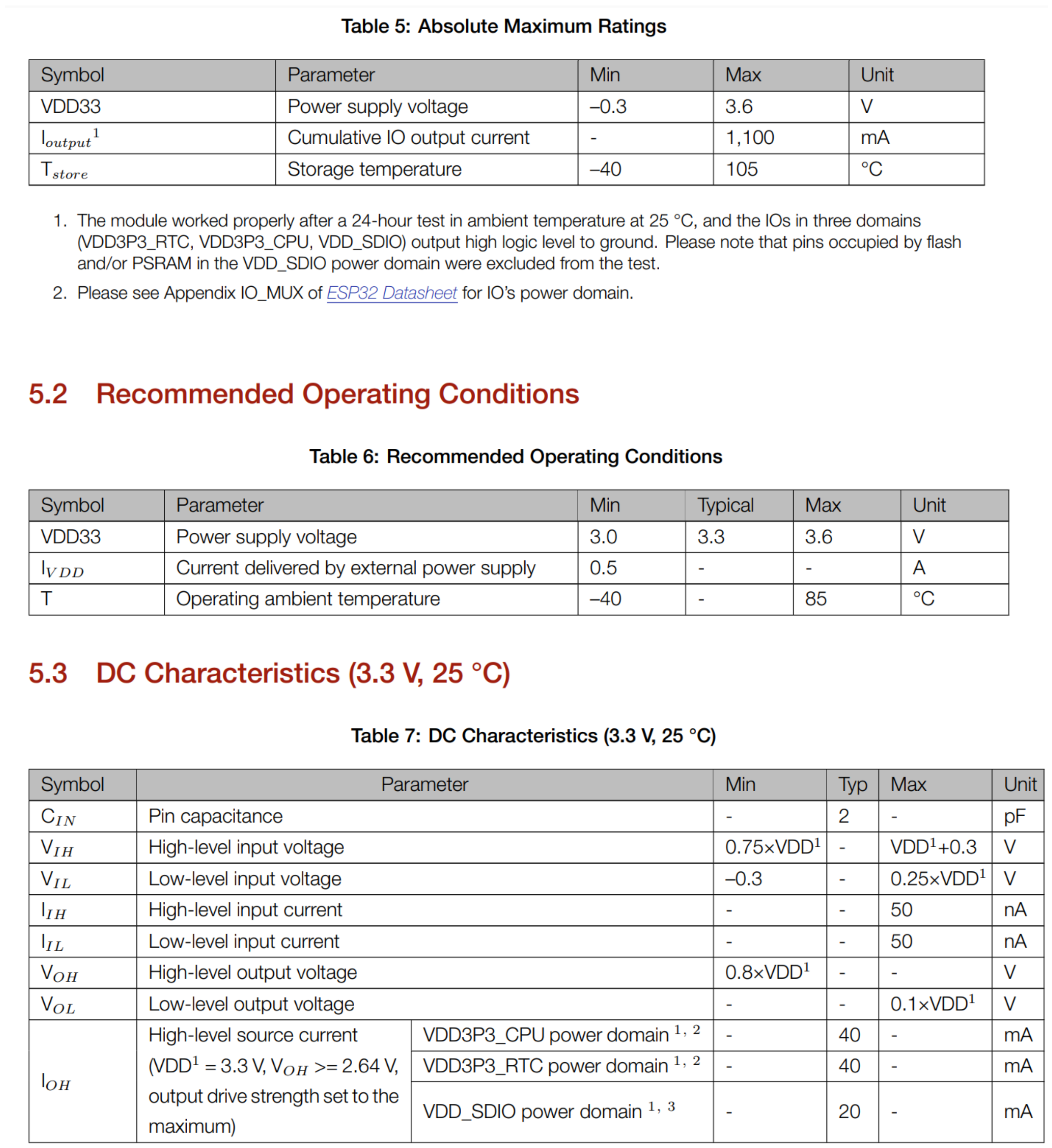


Table 5: Absolute Maximum Ratings

Symbol	Parameter	Min	Max	Unit
VDD33	Power supply voltage	–0.3	3.6	V
I_{output}[1]	Cumulative IO output current	-	1,100	mA
T_{store}	Storage temperature	–40	105	°C

1. The module worked properly after a 24-hour test in ambient temperature at 25 °C, and the IOs in three domains (VDD3P3_RTC, VDD3P3_CPU, VDD_SDIO) output high logic level to ground. Please note that pins occupied by flash and/or PSRAM in the VDD_SDIO power domain were excluded from the test.
2. Please see Appendix IO_MUX of ESP32 Datasheet for IO's power domain.

5.2 Recommended Operating Conditions

Table 6: Recommended Operating Conditions

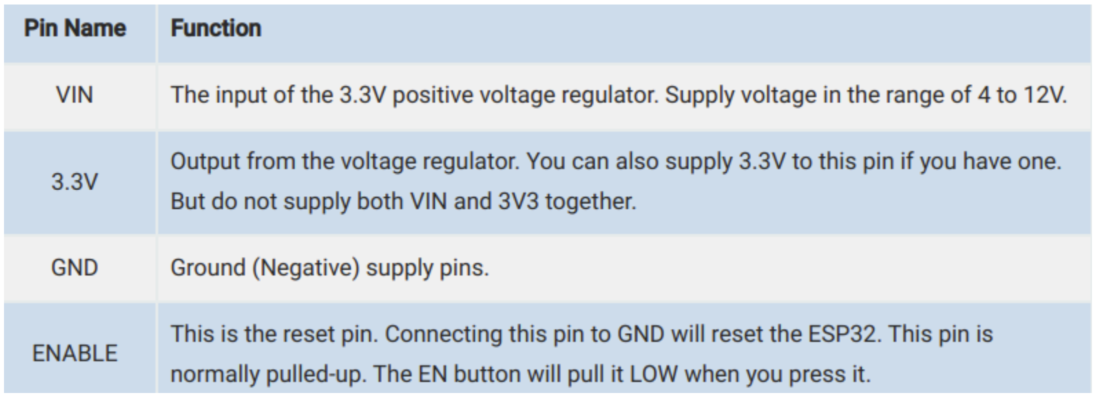
Symbol	Parameter	Min	Typical	Max	Unit
VDD33	Power supply voltage	3.0	3.3	3.6	V
I_{VDD}	Current delivered by external power supply	0.5	-	-	A
T	Operating ambient temperature	–40	-	85	°C

5.3 DC Characteristics (3.3 V, 25 °C)

Table 7: DC Characteristics (3.3 V, 25 °C)

Symbol	Parameter		Min	Typ	Max	Unit
C_{IN}	Pin capacitance		-	2	-	pF
V_{IH}	High-level input voltage		0.75×VDD[1]	-	VDD[1]+0.3	V
V_{IL}	Low-level input voltage		–0.3	-	0.25×VDD[1]	V
I_{IH}	High-level input current		-	-	50	nA
I_{IL}	Low-level input current		-	-	50	nA
V_{OH}	High-level output voltage		0.8×VDD[1]	-	-	V
V_{OL}	Low-level output voltage		-	-	0.1×VDD[1]	V
I_{OH}	High-level source current (VDD[1] = 3.3 V, V_{OH} >= 2.64 V, output drive strength set to the maximum)	VDD3P3_CPU power domain [1, 2]	-	40	-	mA
		VDD3P3_RTC power domain [1, 2]	-	40	-	mA
		VDD_SDIO power domain [1, 3]	-	20	-	mA

Fig. X. ESP32-WROOM-32 Chip DC Output [2]



Pin Name	Function
VIN	The input of the 3.3V positive voltage regulator. Supply voltage in the range of 4 to 12V.
3.3V	Output from the voltage regulator. You can also supply 3.3V to this pin if you have one. But do not supply both VIN and 3V3 together.
GND	Ground (Negative) supply pins.
ENABLE	This is the reset pin. Connecting this pin to GND will reset the ESP32. This pin is normally pulled-up. The EN button will pull it LOW when you press it.

Fig. X. ESP32-WROOM-32 Module DC Supply [1]

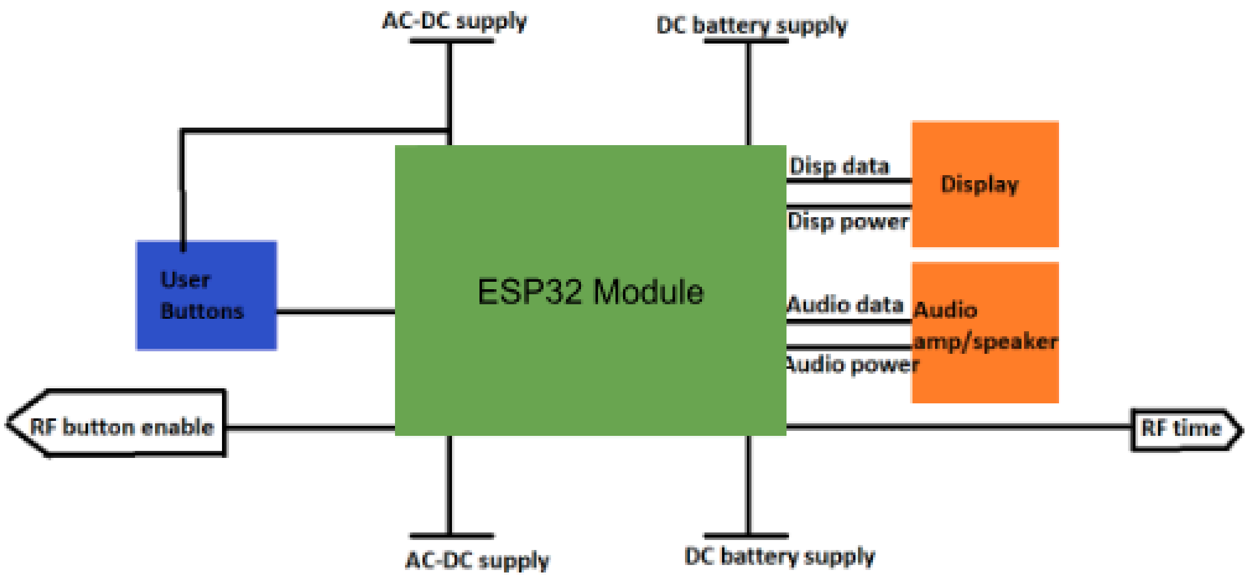


Fig. X. ESP32 Module Layout

3. General Validation

The reason that a microcontroller in general was chosen to communicate between all of the components of the alarm clock was that they are quite general purpose and can be utilized in a multitude of ways. The ESP32 was chosen in this role as the main chip due to its versatile input and output setup. It can communicate with the components in SPI, I2C, PWM, and WiFi, just as needed in this project. The display requires SPI and PWM, the speaker requires I2C, and the web server requires WiFi to communicate with.The power output that is capable for the microcontroller is also sufficient for the other components of the project, and can also simply share the load its power source as well.

The compatibility that the ESP32 has with the Arduino framework also provided much software support and the necessary libraries and development environment to complete the product. The community support and content that revolves around the Arduino is something that is hard to compete with, allowing the project to incorporate clean and smooth incorporation of other creator made libraries to work alongside the custom current project based libraries simultaneously. Thus, completing the proper desired functionality, interconnecting the other components into a fully functional system, capable of accomplishing the promised tasks.

The pinout that the project is designed around is available, and the abundance of pins allows the components attached to the microcontroller to not cause any conflicting needs. This versatility of options is also accompanied by the microcontroller module’s ability to take in a variety of voltages. While the chip itself is sustained by a specific voltage of 3.3V, the module incorporates a regulator system to allow voltage to range from 4-12 volts. That way, our battery backup system component can supply the configured voltage of 5 volts.

The code that will be associated with this block will be incorporated into different modular libraries, that each component will be sectioned into. That way the overall code development will be much cleaner, useful for future improvements and quick to fix or change. The reason this is possible is due to the openness of the Arduino software that the ESP32 utilizes, allowing the users to write their own C++ libraries and including them into the flashed program into the ESP32 chip. We chose this framework because it also is useful in dividing labor into each of the developers, creating a simple way to program, while having no dependency on other blocks, creating a versatile software layout. The logic will be simpler overall during integration. The high efficiency and low amount of power draw that the ESP32 will draw is also useful, as the goal of the project is to maintain efficiency and create a reliable backup power source that can maintain nominal function in a non-external power situation.

4. Interface Validation

Interface Property	**Why is this interface this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
alrm_st_ntrfc_mcrcntrllr_rf : Input		
Messages: The transmitted data is the time of alarm that is to be set.	The message is simply to indicate successful transmission to the user.	The system is implemented meets such, and verification was completed successfully.
Messages: A message transmitted is the enable/disable for the alarm reset button	The message is simply to indicate successful transmission to the user.	The system is implemented meets such, and verification was completed successfully.

Other: A confirmation message is displayed to the screen once the information of the alarm alarm is successfully sent to the microcontroller	The message is simply to indicate successful transmission to the user.	The system is implemented meets such, and verification was completed successfully.

usr_bttns_mcrcntrllr_dsig : Input

Logic-Level: Active HIgh	This signal will be used to translate the buttons state	The ESP32 Data sheet Pg15-16 [2] displays the voltage for sensing and output to fit 3.3V
Vmax: 3.6V	3.3 is the nominal, fitting under the 3.6V window	The ESP32 Data sheet Pg15-16 [2] displays the voltage for sensing and output to fit 3.3V
Vmin: 2.5V	3.3 is the nominal, fitting above the 2.5V window	The ESP32 Data sheet Pg15-16 [2] displays the voltage for sensing and output to fit 3.3V

bckp_bttry_chrgr_mcrcntrllr_dcpwr : Input

Inominal: 102mA +- 2mA	The calculated summation current draw for nominal operation	The load was tested and verified to not exceed, during load conditions.
Ipeak: 104mA +- 2mA	The calculated summation current draw for nominal operation	The load was tested and verified to not exceed, during load conditions.
Vmax: 5.25V	The microcontroller can handle 4-12V, fitting within the range.	The ESP32 Module datapage [1] shown in the design figure, explains the range fits of voltage input ranges 4-12V
Vmin: 4.75	The microcontroller can handle 4-12V, fitting within the range.	The ESP32 Module datapage [1] shown in the design figure, explains the range fits of voltage input ranges 4-12V

mcrcntrllr_dsply__dsig : Output

Logic-Level: Bit value of 0 or 1	This is a simple PWM method to communicate	The method is tested and is verified.

Max Frequency: 1Hz	The PWM frequency is based on the microcontroller's maximum.	The screen display Pg 176 [3] shows the frequency as compatible. And the ESP32 Pg 1 [2]
Vmax: 2.4V	This signal is set based on the display signal input voltage	The ESP32 [2] shows the voltage as compatible.

mcrcntrllr_dsply__dcpwr : Output

Inominal: 68mA	Calculated from display data sheet and nominal draw	The nominal was tested and verified even after estimating checking the voltage rating on the ESP32 data sheet [2]
Ipeak: 83mA	The display datasheet tells the peak ratings	The peak was tested and verified even after checking the voltage rating on the ESP32 data sheet [2]
Vmax: 5.0V	This signal is set based on the display's input supply voltage	The peak was tested and verified even after checking the voltage rating on the ESP32 data sheet [2]
Vmin: 3.3V	This signal is set based on the display input supply voltage	The minimum was tested and verified even after checking the voltage rating on the ESP32 data sheet [2]

mcrcntrllr_usr_bttns_dsig : Output

Logic-Level: High	This signal will be used to translate the buttons state	The ESP32 Data sheet Pg15-16 [2] displays the voltage for sensing and output to fit 3.3V
Vmax: 3.6V	3.3 is the nominal, fitting under the 3.6V window	The ESP32 Data sheet Pg15-16 [2] displays the voltage for sensing and output to fit 3.3V
Vmin: 2.5V	3.3 is the nominal, fitting above the 2.5V window	The ESP32 Data sheet Pg15-16 [2] displays the voltage for sensing and output to fit 3.3V

mcrcntrllr_spkr_dsig : Output

Logic-Level: Active High (I2S)	The speaker demands the protocol.	The ESP32 module has I/O pins for such operation [1]
Other: Digital Low (Max Value): 0.6V	The speaker requires such low voltage for low signal	The ESP32 Data sheet Pg15-16 [2] displays the voltage for sensing and output low fit under 0.6V (0V)
Other: Digital High (Min Value): 1.3V	The speaker requires such low voltage for high signal	The ESP32 Data sheet Pg15-16 [2] displays the voltage for sensing and output high to fit above 1.3 (3.3V)

mcrcntrllr_spkr_dcpwr : Output

Inominal: 2mA	The speaker draw demands no more than such during nominal operation.	The nominal was tested and verified even after estimating checking the voltage rating on the ESP32 data sheet [2]
Ipeak: 20mA	The speaker draw demands no more than such	The peak was tested and verified even after checking the voltage rating on the ESP32 data sheet [2]
Vmax: 5.5V	The speaker draw demands no more than such during nominal operation.	The peak was tested and verified even after checking the voltage rating on the ESP32 data sheet [2]
Vmin: 2.5V	The speaker draw demands no less than such during nominal operation.	The minimum was tested and verified even after checking the voltage rating on the ESP32 data sheet [2]

5. Verification Plan

1. First verify the ESP32 is receiving power, as the LED should maintain illumination.
2. Unplug the external ac-dc source
3. Verify that battery is supplying to the ESP32 as well
4. Next the display interactions will be tested in order to properly verify the next few components. This is done by hard coding the display to output a certain number to display.
5. Check if the number is properly displayed as hard coded.

6. When complete, verify button interactions by only outputting on the display when a button is pressed.
7. Do this for each button
8. Check if the audio is properly playing, thus hardcode a signal to be output into the speaker.
9. Connect the the wifi, and verify connection stability
10. Continuously send signals the the web interface
11. Verify via web interface that the signal is being received.
12. Send data from the web interface to the board and display that information on the graphical display.
13. Verify that the same information is being displayed on the the display

6. References and File Links

[1] VISHNU MOHANAN, "DOIT ESP32 DevKit V1 Wi-Fi Development Board – Pinout Diagram & Arduino Reference - CIRCUITSTATE Electronics," *CircuitState*, Dec. 20, 2022. https://www.circuitstate.com/pinouts/doit-esp32-devkit-v1-wifi-development-board-pinout-diagram-and-reference/

[2] "ESP32-WROOM-32 Datasheet." Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[3] A. Industries, "2.8" TFT LCD with Touchscreen Breakout Board w/MicroSD Socket," *www.adafruit.com*. https://www.adafruit.com/product/1770

[4] "Pololu - Wall Power Adapter: 5VDC, 1A, 5.5×2.1mm Barrel Jack, Center-Positive," *www.pololu.com*. https://www.pololu.com/product/1469

[5] "MCP73831/2 Features." Available: https://ww1.microchip.com/downloads/en/DeviceDoc/20001984g.pdf

[6] "Rev. 1. 6 BCD Semiconductor Manufacturing Limited Data Sheet General Description," 2010. Available: https://www.diodes.com/assets/Datasheets/AP3012.pdf

[7] "LM4875 750 mW Audio Power Amplifier with DC Volume Control and Headphone Switch Check for Samples: LM4875 1FEATURES DESCRIPTION TYPICAL APPLICATION CONNECTION DIAGRAM Small Outline Package (SOIC) Mini Small Outline Package (VSSOP) Top View See Package Number D, DGK Figure 1. Typical Audio Amplifier Application Circuit." Accessed: Mar. 08, 2024. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm4875.pdf

[8] jconrow00, “IOT-Alarm-Clock/Website Interface at master · jconrow00/IOT-Alarm-Clock,” *GitHub*, 2024. https://github.com/jconrow00/IOT-Alarm-Clock/tree/master/Website%20Interface (accessed Mar. 09, 2024).

[9] “LM4875 BOOMER 750mW Audio Pwr Amp w/DC Vol Contro & Headphone Switch datasheet (Rev. C),” *ti.com*, May 2013. https://www.ti.com/lit/ds/symlink/lm4875.pdf (accessed Mar. 09, 2024).

[10] “135694,” *jameco.com*, Jul. 29, 2021. https://www.jameco.com/Jameco/Products/ProdDS/135694.pdf

7. Revision Table

3/4/24	Joseph Conrow: Revamped entire block validation
1/26/24	Joseph Conrow: Revision with final product decision specs
12/7/23	Joseph Conrow: Initial content creation

8. Revision Statement

The first few revisions of the validation resort created on this block had mostly to do with speculations on decided module block specifications. The final products were not yet confirmed, yet now, the components to be used in conjunction with this block are finalized and the information to be understood and validated and verified are known with confidence. By designing and providing interface information in reference to these such blocks, the full description and validation can be fully confirmed. The entire validation for the block was redone, with retrospect to knowledge gained from an already completed verification as well. Thus, making it absolutely sure of the validity of such properties, and such designs. The instructor provided good feedback such as pinouts to provide in this document. The other team members also provided helpful information for the properties of the interfaces between this block and their blocks. This is surly guided by the information of the datasheet and proof of verification. The interfaces were all changed since last revision, and most of the diagrams in the design section were as well. The description has evolved alongside the general validation section. This validation was done with much foresight as the team completed the block verifications of the project system already, which is often nonsensical or non-practical rather.