Project Writeup

What is this project?

This project aims to create an IoT disinfecting system using RT-Thread which harnesses recent developments and proliferation of UV-C LEDs which offer a low-cost alternative to mercury-vapor lamps in disinfecting. The project uses low-cost and widely available components which can be sourced and used anywhere. The intended environment of the project is areas which humans occupy and may transmit bacteria, viruses or fungi spores to other humans. In this case, UV-C light provides a safe, low-cost and maintenance-free method for disinfecting areas where germs gather. An important design goal of the project was to make it highly modular and made up of common parts. Very often, the people who need such a low-cost disinfecting device may be in areas which lack all but the most common parts, and many of the design choices made are influenced by the prioritization of such cases.

The project uses a client-server architecture for each room. Both the client and the server are controlled by Raspberry Pi Pico modules (due to their availability and low cost globally). Due to the usage of RT-Thread, the code can easily be adapted to work on any other module or microcontroller with the same basic capabilities. The server's purpose is to verify that there are no humans present which could be impacted by UV-C light, while also having a set time when the UV-C light can be active. This ensures that no humans can be impacted by UV-C light, which is harmful after a long exposure. When all conditions are met, the server signals to the clients that they can disinfect their area using a simple set of commands. The clients are the devices with the UV-C LED and they turn the LEDs on for a precisely calculated time which ensures that most of the aforementioned germs are eliminated.

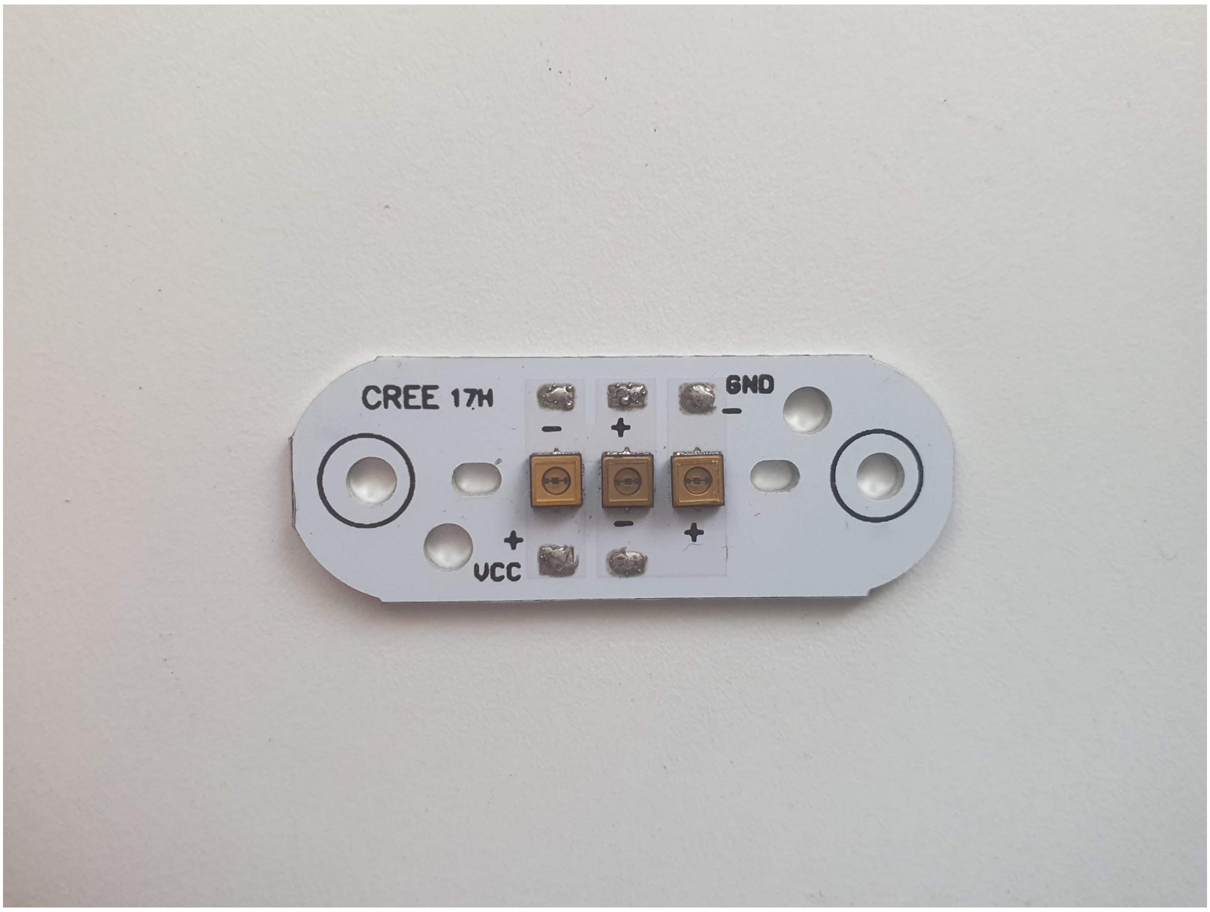
The server uses a microwave radar to sense the presence of humans (or animals) in the vicinity. Even though this device only operates on movement, given direct line-of-sight (such as when placed on the roof of a room) even breathing is enough to set it off. If it detects a human, the microwave radar signals all clients to turn off their LEDs immediately. Furthermore, all clients have a visible-light LED and a buzzer to indicate that they are on, so that humans in the vicinity are notified that the room is being disinfected.

From the beginning the project was designed to be able to accommodate a multitude of wireless and wired communications protocols. The given example uses two ESP32 microcontrollers with the ESP-NOW wireless protocol acting as UART bridges, however the project can be adapted to use many different protocols due to the usage of RT-Thread's UART Module. It was tested to work with RS232, eschewing the need to use wireless communication if it is not necessary or a wired connection is desired due to greater noise immunity and reliability.

An important positive side to this project is the low maintenance needed to keep the system functional and operational. Apart from the initial setup and potentially recharging the batteries, no active human intervention is needed. This eliminates the need for someone to repeatedly disinfect surfaces if the project is used in an environment frequented by many people.

Why Use UV-C?

UV-C light, a powerful tool in the battle against harmful bacteria and microorganisms, has gained significant attention in recent years due to the COVID-19 pandemic. With the advent of low-cost UV-C LED technology, this fascinating light has become more accessible and efficient. In the following sections I will describe the massive benefits of this technology and why I used it in my project.



The specific model of UV-C LED I chose (can be connected in parallel for more light)

The Power of UV-C Light

UV-C light, specifically in the wavelength range of 200 to 280 nanometers (nm), possesses exceptional germicidal properties. It targets the DNA and RNA of bacteria, viruses, and other pathogens, disrupting their genetic material and rendering them unable to reproduce or cause harm. This process, known as photoinactivation, effectively neutralizes the microorganisms, preventing their proliferation and spread. (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8238411/)

UV-C LEDs: A Technological Breakthrough

The introduction of UV-C LEDs has revolutionized the utilization of UV-C light for disinfection purposes. These compact and energy-efficient devices offer several advantages over traditional UV-C lamps, making them an ideal choice for various applications.

1. Precise Emission: UV-C LEDs are specialized to emit light in a narrow wavelength range which varies for each LED. This allows the systems integrator to target particular bacteria and viruses which may be more vulnerable to specific wavelengths, which ensures maximum disinfection efficiency, minimizing energy waste.

2. Instant On/Off: Unlike conventional UV-C lamps that require warm-up and cool-down periods, UV-C LEDs provide instant illumination. This feature allows for greater flexibility in implementing disinfection cycles and enables rapid response in critical situations.

3. Longer Lifespan: UV-C LEDs have an extended operational lifespan compared to traditional lamps. With proper use and maintenance, these LEDs can provide reliable and consistent germicidal performance for thousands of hours, reducing the need for frequent replacements. Most LEDs last between 4000-10000 hours of use, which when combined with the calculations below

4. Compact Size: UV-C LEDs are compact and lightweight, allowing for easy integration into various devices and systems. From handheld sterilizers to HVAC systems, their small form factor enables versatile and scalable applications across diverse industries.

Benefits of UV-C LED Implementation

The utilization of UV-C LEDs for bacteria eradication offers numerous advantages:

1. Enhanced Safety: UV-C LEDs emit light within a controlled range, minimizing the risk of exposure to harmful radiation. Compared to other disinfection methods that involve chemicals or high temperatures, UV-C LED technology provides a safer alternative for bacteria elimination.

2. Environmentally Friendly: UV-C LEDs are energy-efficient and do not require the use of hazardous chemicals like mercury, used in traditional lamps.

3. Cost-Effective: Although UV-C LEDs initially involve a higher upfront investment, their long lifespan and lower energy consumption contribute to significant cost savings in the long run. Additionally, the reduced need for chemical disinfectants translates to fewer ongoing expenses.

4. Versatility: UV-C LEDs can be integrated into various devices and systems, making them suitable for a wide range of applications. From healthcare facilities and food processing plants to homes and public spaces, UV-C LED

technology provides effective disinfection wherever it is needed.

Technical details of the implementation

UV-C light, harnessed through the power of UV-C LEDs, presents a compelling solution for bacteria eradication. With its proven germicidal properties, compact design, and numerous benefits, UV-C LED technology offers a safer, more efficient, and environmentally conscious approach to disinfection. In my project, I chose to use 265nm UV-C LEDs as that region of wavelengths has been proven to be the most efficient in eliminating bacteria and viruses (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8238411/).

To calculate the length during which the UV-C LED needs to be on, first a few constants need to be known. Most UV-C LEDs have a relatively low efficiency which varies according to manufacturer. Contacting my manufacturer, I received a UV-C LED power value of 15.1mW of UV-C light output per 100mA at a Vf = 6.2V which was measured using specialized equipment. From this, the intensity of the light output from the LED can be calculated since this project aims to run the LED at those identical specifications for forward voltage and current.

Most bacteria, viruses and spores require a total of 8mJ/cm^2 to be destroyed. ([LEDs for sterilization with UV-C light | Electronic components. Distributor, online shop – Transfer Multisort Elektronik (tme.eu)]) Since the LED outputs 15.1mW, its intensity on a 1m^2 surface is 15.1mW/m^2 or 1.51µW/cm^2. The time calculation from here is simple:

$$time = \frac{8mJ/cm^2}{1.51\mu W/cm2} = \frac{8*10^{-3}}{1.5*10^{-6}}s = 5333.3s = 88.9min = \sim 1.5h$$

When combined with the 4000-10000 hour lifespan of UV-C LEDs, this results in a 7-18 years of daily operation without any need for maintenance given all other components last as long. Since the UV-C LED is a relatively new technology, this shows the significant advantage it has over older technologies in terms of reliability, which is enhanced by the circuit.

Harm mitigation

Despite the safety features included in the project, it is still not completely safe. Even though the power at which the LEDs operate at is unlikely to cause much harm to humans, it is important to note that the project has not undergone any rigorous official standards conformance testing. As such, it is only recommended to build and implement it for personal use, with full acknowledgment of the risks involved. The critical weakness of the system is if the Doppler radar module fails, which the server will detect as a persistent lack of humans. At the specified times, it will then turn on even if there are humans present. As such it is imperative that the Doppler radar module is tested to work, or a PIR sensor can be added to complement it (as is common in many commercial human-presence detectors).

Explaining the choice of power

A key consideration for this project was the source of power used. Since the UV-C LED and the microcontroller system were designed so that anyone could make them, even with very limited resources, I chose to design in a power bank instead of a discrete battery with a charging circuit. Since the main microcontroller board (the Raspberry Pi Pico), as well as most other commercial boards have USB connectors, I chose to use the Pico's USB connector as the main power source, which means that the project can either be powered by a phone charger or power bank. For the clients, I prefer the power bank option. There are multiple reasons for this, which I will describe below:

1. Power banks are highly cost-optimized due to the massive scale they are built at, making them a cost-effective solution for projects of various sizes and budgets. Additionally, their widespread availability in the global market ensures easy access to power banks, allowing people to source them conveniently for their projects.

2. Power banks provide a portable and self-contained power solution. Unlike a battery with a charging circuit, which requires additional components and wiring, a power bank offers a compact and integrated package. This eliminates the need for complex circuit design and simplifies the overall system architecture.

3. Power banks are designed to be rechargeable, eliminating the hassle of constantly replacing batteries. With a power bank, the device can easily be recharged using a standard and common USB charger, ensuring a continuous and reliable power supply for the project. Depending on the use case, the power bank can act as a small UPS which keeps the device working while a charger is permanently connected to it.

4. Power banks usually incorporate built-in protection mechanisms such as overcharge protection, overcurrent protection, and short-circuit protection. These safety features safeguard both the project and the connected devices, reducing the risk of damage or accidents. Designing in these features is difficult and expensive, often requiring specialized ICs in unusual packages which can be difficult to obtain.

Overall, the convenience, portability, versatility, and built-in safety features offered by power banks make them a preferred choice for this project. They simplify project development, enhance flexibility, and provide a reliable power source, which allows me to focus on the core aspects of my design without worrying about power management.

Explaining the choice of communication

In the project I designed the Wi-Fi module (ESP32) to be separate from the main microcontroller which turns the UV-C LED on and off while providing monitoring, as well as the main server which monitors the environment. There are a few reasons for this:

1. Flexibility and scalability: By using a separate main microcontroller and Wi-Fi module, anyone building the project can have more flexibility in choosing the specific microcontroller that best suits the project's requirements. This allows for better scalability and future-proofing, as the main microcontroller can be upgraded or replaced independently without affecting the Wi-Fi functionality. When combined with the flexibility of RT-Thread and the use of RT-Thread built-in functions in the code, this means that practically any supported microcontroller with the required features can work immediately with the project code.

2. Flexibility in communication protocol choice: In case the ESP32 is not available, the same code will work with an ESP8266, or even changed to use Bluetooth, ZigBee or another communications system. In this project, the ESP32 is simply used as a common stand-in for a wireless communications module, and is meant to be replaceable without having an effect on the system as a whole. Due to the fact that the UART protocol was used with the ESP32, anything that communicates over UART can be used without modification. It is even possible to control it with RS485 or RS232, which would make the system more reliable, while removing the wireless requirement.

3. Modularity and modifiability: Separating the main microcontroller and Wi-Fi module promotes modularity in the system design. It allows for easier modification or upgrade of specific components without affecting the entire system. This flexibility can be advantageous when integrating different communication protocols, adding new features, or adapting the system to changing requirements.

Specific model choice

In this project I specifically chose the ESP32-C3 due to the fact that it has the lowest cost out of the whole ESP32-C3 lineup (at the time of writing this). It has less RAM and a lower processor frequency than the original, however for this purpose that is not relevant, since it operates only periodically, sending low quantities of data. Also, importantly, the less RAM and lower frequency result in a lower power consumption figure for the circuit, which is relevant, given that it does not operate for most of the day.

For the purpose of simplicity, since the ESP32-C3 is not the star of the project and I want the sketch to be usable with all other types of ESP microcontroller, I used the Arduino framework to orchestrate the ESP-NOW protocol communication. The ESP32-C3 has a specific procedure for uploading code. In the Arduino IDE, in the section named Tools, first go to USB CDC On Boot and set it to Enabled. When uploading, hold the BOOT button before you connect your ESP32-C3 and upload your sketch while holding it. Afterwards, to turn it on, simply press EN.

Explaining the LED driving circuitry

There exist countless ways to drive an LED which vary in the complexity of parts required. This project has two features which complicate the process of driving the LED, which are the 4.5-5.5V USB power and the ~6.5V UV-C LED forward voltage. This makes it impossible to drive the LED from the given supply alone, which limits my options to some kind of boost converter. In the end, I chose the ubiquitous MT3608 boost converter module due to the fact that it is low-cost and has an adjustable output voltage.

Tuning

To limit the current, I chose to use a low-value resistor which can be tuned together with the MT3608 to produce the desired output current. The same calculations listed in the UV-C section can be changed with any desired current. However, it is important to know that UV-C LEDs are somewhat sensitive to overcurrent, so be careful when tuning the boost converter. I recommend building the circuit on a breadboard first. Start with connecting the boost converter to the LED through the resistor. Turn the tuning potentiometer on the board to the lowest voltage and slowly increase it. You can connect your multimeter in series in current-measurement mode to set your desired current. Make sure to be mindful of both the maximum current of your LED and the maximum power dissipation of your resistor. The formula $P = I^2R$ is all you need to make sure that you are safe within the limit. For my project, I chose 12 ohms for my resistor since I wanted to drive the LED with 100mA of current and not waste too much current. I will outline why I chose these numbers below.

LED driving calculations

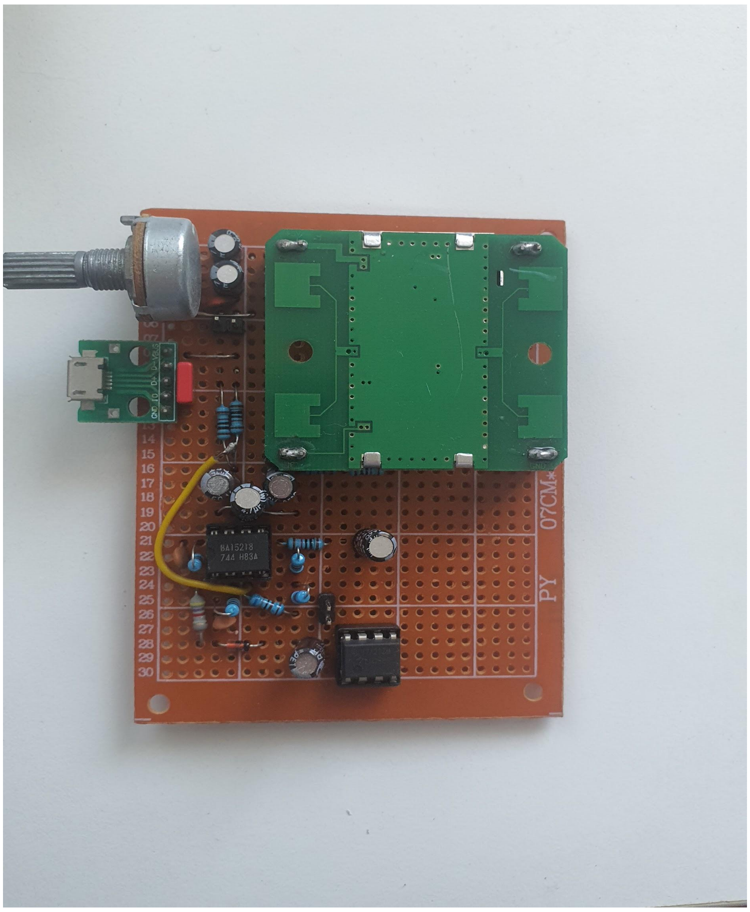
To be able to turn the LED on and off, I chose to use the common NPN BJT 2N3904, since it can handle the currents used to drive UV-C LEDs (max 625mA). For this purpose, the vast majority of NPN transistors available on the market can be substituted. The 2N2222 or BC547 are both good and widely available substitutes. Since BJT transistors have a voltage drop between collector and emitter (~300mV), this is an additional variable to keep in mind.

Theoretically, if a designer desires a low voltage drop with a resistor, a low resistance can be chosen. However, given the fact that the circuit uses a boost converter to drive the LED, a very low resistance can lead to short overcurrent spikes from the switching. Thus, I would advise against using a very low resistance. For my design I chose that I desired a 1.5V voltage drop after the LED. This meant that I would have to tune my boost converter to ~8V (since the LED drops 6.5V), which would make it more efficient since there is a small difference between 5V and 8V. From the 1.5V, 0.3V will be dropped in the transistor, which means that 1.2V has to be dropped through the resistor. From this, using $V = IR$, the resistor's value will be $\frac{1.2V}{0.1A} = 12\Omega$. If this value is unavailable, the resistor can be substituted with 10 Ohms, keeping in mind to lower the boost converter's voltage to get the appropriate current.

Radar interfacing

The Doppler radar module used in this project (HB100) needs an external circuit to function. This circuit is given in the module's application note and explained in this video: [HB100 Doppler Radar, Limpkin's Amplifier and Arduino MCU – The Details (1/3) - YouTube]. You can substitute the module with a similar radar module. For ease of interfacing, I recommend using the peak detector output, since it outputs an analog voltage proportional to the strength of the received signal. This makes it easy to tune, particularly if you will be using the module in a building, since the signal can pass through walls and be detected by someone outside of the room. I tuned my module manually to detect only movement in my small room, which is represented through the constant ADC_RADAR_THRESHOLD in the code. You can also use the frequency output, disregarding the velocity component.

You can also choose to use any other radar module with similar functionality, as the same functional principles apply to all radar modules of this type. You can mount the radar in plastic or wooden boxes and it will not block the radar signal. For the project, I chose to test it out in a small plastic project box, and the server box was completely closed. This means that the project can also be used in environments with a lot of dust or moisture without fear that it will damage the system!



I built the radar analog board externally so that it can be reused

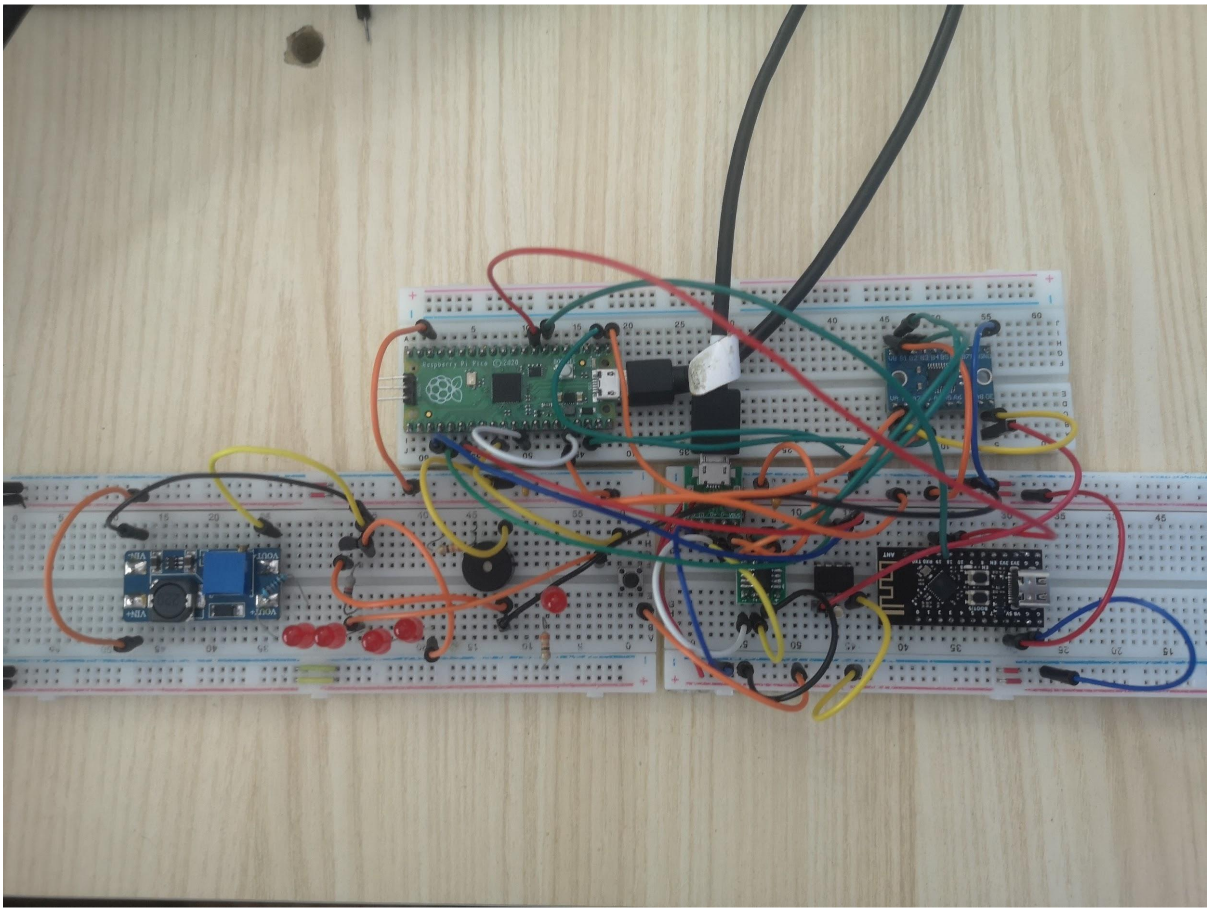
Part selection and Bill of Materials (BOM)

The bill of materials is included in the project directory in a format that LCSC will accept, which means that you can easily get most of the materials you need. Most of the parts can be substituted, as was discussed previously. The microcontroller boards can be replaced with anything that supports RT-Thread. The LED driving circuitry can similarly use practically any adjustable boost converter, as well as any (rational) combination of NPN transistor with resistor. To reduce the parts count by removing the BJT base transistor, an N-Channel MOSFET can also be used.

The operational amplifier selection is somewhat important. In the schematic, I chose an LM358, since even though it is regarded as a low-end operational amplifier, it has one essential feature in this project - the output can go down to 0V. This, along with a low input offset voltage are the two features you should look for when selecting a substitute operational amplifier. In particular, you might have difficulty with getting a TL072 or similar to work due to it being unable to go down to the negative (0V) rail.

You can choose to add a CH340N (or any other CH340) breakout board to the Raspberry Pi Picos to control them directly from a computer. The code includes RT-Thread MSH functions for debugging and manually enabling the UV-C LEDs. If you want to use the system as intended and without debugging, you can omit the CH340s.

For further BOM consolidation, you can substitute the 10uF capacitors with 4.7uF since the values are not critical, as well as replacing the 12K resistor with 2x1K + 10K.



The client built on a breadboard with debugging CH340 included. In this photo, for my own safety, I use four red LEDs to simulate the UV-C LED, since the voltage drop is mathematically equivalent.

The clock module is also optional - you can simply add the DS1302 with a 32.768KHz crystal.

I recommend building the project on a perfboard, since there is a relatively low parts count and a PCB would come out somewhat expensive due to the large size of the radar module + Pico + ESP32.

The only part which might be difficult to obtain in this project is the UV-C LED. For those, I recommend using online retailers like Aliexpress and eBay. Make sure to look for sellers which give detailed information about the effective power of the LED and its wavelength. If you will be using the LED with a relatively high current, make sure to add a heatsink. For the currents used in the project (6.5V at 100mA resulting in ~650mW) a heatsink is unnecessary, however for larger currents, a heatsink is mandatory.

Software setup and upload

The process for uploading to the ESP32-C3s was outlined in its respective section. For uploading to the Raspberry Pi Pico, simply hold the Boot button and connect it to USB. When you see the file system appear, drag and drop the respective .uf2 file in the Pico.

If you want to modify the code, simply create a project in RT-Thread Studio with your desired board. Then, simply copy the main file to your project. For the server also make sure that you copy the DS1302 library. If your DS1302 is not set up, use the ds1302_set_time function in the code once, then when the time is set, delete it. As long as the DS1302 has a battery or external power, it will keep the time quite accurately.

If you want to change the active time of the LEDs, change the TIME_START and TIME_END (lines 30-31) active constants in the server's main file. Other relevant constants are the ones pertaining to the ADC (lines 23-27) and the LED illumination duration (line 34). The client code similarly has such constants which can be adjusted, notably the pin numbers (lines 21-24).

Conclusion

I believe that this project is both good for learning and practical daily use. The potential applications of UV-C LEDs have not been fully explored, and the technology is constantly seeing improvements. Combined with the power and ubiquity of Doppler radar sensor modules, they can be used in a manner that is safe to humans and animals in the vicinity. Another goal of this project - ubiquity and low cost - was also achieved through the use of available parts and plentiful substitutes. I personally plan on continuing to expand its capabilities as well as contributing to the RT-Thread project, which made this all possible!