ECE399 - B01

Ocean Water Quality Monitoring System

December 15th, 2023

—

Group #2

Teaching Assistant
Khalid Al-Hammuri

Members:
Jenna Hilderman
Brandon Ip
Bruce Lynch
Vlad Maslovski

Table of Contents

Table of Figures

Table of Tables

Executive Summary

The focus of the project was to design and implement a water quality monitoring system that focuses on affordability and real-time data transmission. The water quality is assessed by measuring key parameters of the water to assess its quality. such as dissolved solid, suspended solids, pH, and temperature. The scope was also to integrate the system with Internet of Things (IoT) technology which allows remote monitoring of the parameters via a graphical dashboard. In response to the growing demand for effective water quality monitoring, this project contributes a low-cost solution, with the aim of keeping the budget under 400$. The inclusion of IoT technology enables real-time data monitoring that facilitates fast decision-making. The design prioritized simplicity and affordability that would allow for the deployment of multiple monitoring stations to cover large areas. The central system components that our system comprised of includes Arduino UNO R4 WiFi microcontroller and sensors that measure dissolved solid, suspended solids, pH, and temperature. All the electronic system components were self-contained in waterproof enclosure. The system is powered using a portable battery pack that can maintain full power for approximately 30.5hrs. The cost of the prototype came under budget at 257.45$. The chosen IoT interface was the Arduino Cloud platform which was used to program and show the graphs of the sensor readings. While the performance of the system in a static testing environment was satisfactory, it is recommended thorough testing is to be conducted in a field environment i.e. in ocean water and exposed to elements.

Introduction

A water quality monitoring device is essential for assessing and maintaining the purity of water sources worldwide, such as municipal water supplies and wastewater treatment facilities. These devices provide real-time data on various water parameters, including chemical composition, microbial content, and physical properties. This data is crucial in allowing governments to make decisions on what course of action is needed to protect water sources from the numerous sources of water pollution, including wastewater from industrial and agricultural sites, oil spills due to accidents involving marine vessels and offshore oil platforms, plastic pollution, and rising levels of acidity and temperature caused by climate change. This project aims to create a water quality

monitoring device that can measure parameters such as the pH level, temperature, and dissolved solids, which serve as measurements of the purification of the tested water. Constraints for this project include the effective range of the water quality monitoring device, the accuracy of the data provided by the sensors, and having an estimated budget of only $200-400.

Literature Review

Introduction

Water is one of Earth's most vital natural resources that affects humans and ecosystems alike. Maintaining water quality for uses such as drinking has become significantly more difficult in the age of growing and expanding industries. For decades, efforts have been made to find different ways to measure quality to ensure that potable water is available for the ever-expanding worldwide population. These methods include boy-based systems, autonomous surface vehicles, and fixed-stations based on shorelines which will be discussed further in this literature review. These solutions will then be compared be to one another to determine which is most suitable for this project.

Buoy-based monitoring system

The buoy-based water quality monitoring system is a floating platform (buoy) with basic hardware of the system including a microcontroller, electrochemical sensors, and a wireless communication system [3]. The sensors are the inputs to the microcontroller and provide information about chemical and physical composition of the water being sampled. These systems typically utilize some type of wireless communication system to transmit logged data to a land-based station. If the buoy is within 15km of a receiver node, a low-power, wide-area network (LPWANs) technology can be used which operates in 902-928 MHz ultra-high frequency range [4]. If the buoy is located close to cellphone towers (<25km), a Global System for Mobile Communications (GSM) can be used which operates in 850-1900MHz radio frequency range [3]. The buoy-based monitoring stations can be integrated with Internet of Things technologies for collection and management of the sensor signals. This allows real-time monitoring and analysis of the collected data [2].

The initial installation of a buoy-based monitoring system can be time consuming and costly if the

system must be deployed in a remote area [2]. The quality and complexity of the communication, microcontroller and sensors can greatly affect the cost of the system. A basic system is generally regarded as low-cost compared to other alternatives [4]. As an example, one basic design was built with NUCLEO-L476RG board with an STM32L476RG Microcontroller, RFM95W transceiver module (low-power wide-area networks communication module), 3 sensors (Temperature, pH, Dissolved Oxygen) for approximately $898.14 (converted from 658 USD) [5]. A much more complex systems can drastically increase the cost. For example, Cleveland Water deployed “smart buoys” in Lake Erie that cost $109,196 each. These buoys are equipped with some sophisticated sensors that monitor things like algae blooms, thermal upwellings, and hypoxia. They also relay marine weather conditions to the land stations by measuring wind and wave parameters [5]. Due to the large cost of these monitoring buoys only a few of them are typically deployed so an advantage of using a low-cost system is that it can be scaled to cover large areas with many buoys [3].

Autonomous Surface Vehicles

Scientific monitoring of marine environments poses a set of challenges; from remote and expansive locations to the potentially dangerous waters that researchers must navigate. In this context, the development of Autonomous Surface Vehicles (ASVs) offers one promising solution to address these challenges. Autonomous Surface Vehicles, also known as Unmanned Surface Vehicles (USVs), are small, unpiloted ocean surface vessels often used for long term or remote data collection.

ASVs present significant advantages over traditional, manually operated vehicles. A notable benefit is the ability to send the vessel into a challenging situation, especially if operation could compromise safety for a human operator [7]. Manual operation off-board requires a human to use a 2.4 GHz remote control to operate this vehicle [7] – which can be done from the safety of a larger, base vessel, or from shore. Regarding automatic operation, an automated Ground Control Station (GCS), can control one or more ASVs. Multiple ASVs send GPS waypoints to the GCS, which returns GPS waypoint, this aids in coordinating multiple vehicles with respect to collision avoidance [8].

The endurance of a research vessel is enhanced with an ASV using solar power [6]. The

photovoltaic measures of the system would largely depend on the specification of other onboard hardware, as well as the area and arrangement of the solar panels [6]. One implementation of this was the ASV known as the Ocean Atmosphere Sensor Integration System (OASIS), and was designed to be a low-cost, reusable, re-configurable, and long duration open ocean observing platform [6]. Along with sensors to aid in navigation and stability, OASIS also housed various atmospheric sensors such as windspeed and direction, barometric pressure, humidity, and temperature. Specific payload sensors and other hardware became customizable and tailorable to research groups [6].

A current system is the Jetyak, designed and built by the University of South Carolina's Autonomous Field Robotics Lab [9]. It is controlled using a Pixhawk PX4 microcontroller, capable of communicating using 900MHz radio modems, 2.4 GHz remote control radios, and a 2.4 GHz Wi-Fi connection [9]. These lower end Wi-Fi transceivers are functional only up to a ~1km range, however, they are less expensive than the Freewave Radios ($3000), which can provide an Automatic Identification System (AIS) of 5-8km range [9]. The cost of components for the Jetyak was less than $15,000 USD [9], however this does not include specific hardware and sensors that would collect data of interest.

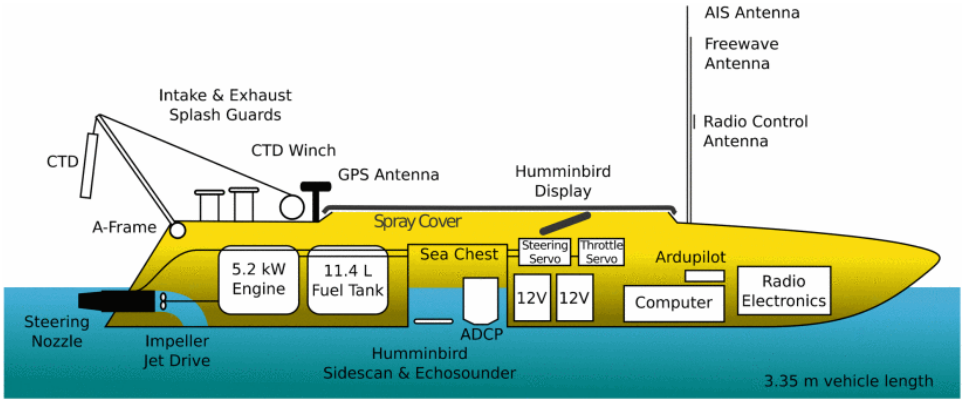


Figure 1: Schematic of components that make up Jetyak [9]. Additional components may be added/substituted depending on experiments.

Fixed-Station Shore-Based Monitoring

Ocean regions local to harbours, mills, and municipalities require careful monitoring of water quality to ensure these infrastructures are not polluting and damaging the local aquaculture. Devising a simple system that would remain fixed to a pier or other permanent fixture could be a solution to long term monitoring of water quality. The device could be self-sufficient by implementing a solar panel with a battery to keep it powered, or it could be run on a power line connected to the local electrical grid. With the permanent nature of the device, it would collect and provide a continuous data set of environmental and water quality variables for a given location – add multiple devices to an area and there could be a spatial degree to the data [10].

An experiment in the River Lee, County Cork, Ireland, implemented a multisensory heterogeneous real-time water monitoring system [10]. The river is comprised of a tidal system, so it was expected to work in fresh- and saltwater. The experiment focused on a few notable features, such as: low cost, robust sensors and a rugged monitoring module, low battery maintenance, and real-time gathering in remote locations [10]. The device was programmed for minimal power consumption with a SLEEP feature, allowing the device to only be ACTIVE for 1% of the time. During the ACTIVE mode, the device was consuming 96mW for 0.039 seconds. Comparatively, during the SLEEP mode, the device was using 0.054mW for 60 seconds [10]. The device also implemented several off the shelf sensors to measure qualities such as water temperature, pH, conductivity, dissolved oxygen, turbidity, and water level. Minimizing power losses is the main point of interest of this experiment and ought to be implemented in future experiments.

The Chesapeake Bay watershed in the United States is home to millions of people and thousands of species, thus careful monitoring of the water quality is of the utmost importance [11]. For an experiment, fixed sensors were developed for optimum sampling sites, and could give spatial temporal data from a GIS map. The experiment selected a single sensor, YSI 6820, which could measure water conductivity, temperature, pH, dissolved oxygen, and several others relating to other parameters of interest for their experiments. The criteria for choosing this sensor were maintainability, operational quality, and reliability, and was weighed against other similar sensors [11]. Choosing a fixed water quality monitoring system allows one to focus on a high quality, low maintenance system and avoid unnecessary complications regarding mobility.

Solution Comparison

Each solution aids in water quality analysis in a different manner. A buoy system allows remote sensing in a constant location offshore. However, being offshore presents issues for unexpected maintenance, consistent and reliable power, and data transmission. While it is possible to mitigate these issues, they are difficult to remove entirely. Similarly, an Autonomous Surface Vehicle meets the same challenges as the buoy. As well, the ASV has moving parts; its motor and rudder are at risk for damages, causing the ASV to potentially be cast astray. Furthermore, the ASV requires guidance and navigation components and software, a robust and seaworthy hull, and a powerful battery to support extended excursions – adding to the cost. Note, however, a single ASV can cover vast areas of the ocean, potentially doing the same work of many buoys. The simplest solution, the fixed station, can be easily set up, maintained, and requires no means of floatation, or mobility – thus it is the cheapest. The primary drawback of the fixed station is that it mainly analyzes shore water quality and not open ocean. However, due to the ease of implementation, the fixed station will be the main model for this project so that water quality monitoring may remain the focus.

Conclusion

This review discusses three solutions: buoy-based monitoring systems, autonomous surface vehicles (ASVs), and fixed-station shore-based monitoring. Buoy-based monitoring systems offer advantages such as long-term operation through solar power, real-time data transmission, and scalability for large areas. Autonomous Surface Vehicles (ASVs) provide an innovative solution for marine data collection and extended endurance through solar power. Fixed-station shore-based monitoring systems can be self-sufficient through solar panels or connected to the local electrical grid, providing continuous data sets for a specific location. Each method has advantages and disadvantages, making its suitability dependent on individual projects. When choosing a monitoring system, one must consider cost, location, maintenance needs, and data collection goals.

Requirement Specifications

The description of desired system features and interface requirements, assumptions, and constraints is outlined in this section. The goal is to build a floating, low-cost (under 400$) water quality monitoring station that is integrated with Internet of Things technology. The system must communicate gathered data which is then displayed on a website dashboard that can be viewed by anyone with an internet connection and a web browser.

Desired System Features

Microcontroller:

Input Voltage: 6-24VDC
Processor: Renesas RA4M1 (Arm® Cortex®-M4) or better
Clock Speed: At least 48MHz
Memory: Minimum 256kB Flash, 32kB RAM
Analog Input Pins: minimum 4
Digital I/O Pins: minimum 2
Communication: 2.4GHz Wi-Fi with 40MHz bandwidth, minimum 45-meter range, power consumption of maximum 0.75W

Water Quality Sensors:

Parameters: total dissolved solids(gravity), total suspended solids(turbidity), pH, temperature
Response Time: <1s, (<1min for pH)
Accuracy: ±5% (± 0.5 degrees for temperature)
Analog output: 0-5V
Protection: sensors must be waterproof
Power Consumption: 40mA max

Data Storage:

5GB cloud or minimum 90day data retention

Energy System:

Expected Lifetime: minimum 300 charge cycles.
Power Consumption: ~1.5W for current system specifications
Recharge Frequency: 24hrs
Battery Size: minimum 26Whr

Interface Requirements

- Website dashboard with 4 graphs displaying real-time data of total dissolved solids(gravity), total suspended solids(turbidity), pH, temperature VS. time
- Accessible via website link to anyone with an internet and a browser

Assumptions

- Demonstration will be performed within WiFi range.
- Normal operation of the system will be in a wet environment.
- Power supply will be stable

Constraints

- Low-cost system with a maximum total budget of 500$
- Project must be completed for final presentation within next 42 days

Proposed Design

We have listed the main components of our system design below; the list of sensors is listed in Table 1. The complete datasheets are provided in the appendix.

Arduino UNO R4 WiFi Board

Part #: ABX00087
Input Voltage: 6-24VDC
Processor: Renesas RA4M1 (Arm® Cortex®-M4)
Pins: 14 digital I/O, 6 analog inputs

Portable Power Supply

Part#: W1056
Rated Energy: 38.48Whr
Output: 5V – 3A

Enclosure: 8x8x4 in. waterproof PVC box

Part #: RJB884

Table 1: List of sensors in the proposed design.

Sensor Type	Part#	Rated voltage (V)	Rated Current (mA)	Rated Temperature (°C)
Analog Turbidity	SEN0189	5	30	5 ~ 90
Analog Dissolved Solids	SEN0244	3.3 ~ 5.5	3 ~ 6	N/A
Digital Temperature	DS18B20	3.0 ~ 5.5	4.0	-55 ~ 125
Analog pH	SEN0169-V2	3.3 ~ 5.5	N/A	0~60

The chosen IoT interface for our project is the Arduino IoT cloud because it is designed to work with Arduino microcontrollers. The interface allows as us to create dashboards so that our real-time input data from the sensors can be visualized as a graph. We have constructed the basic flow chart of our system design which is shown in Figure 2.

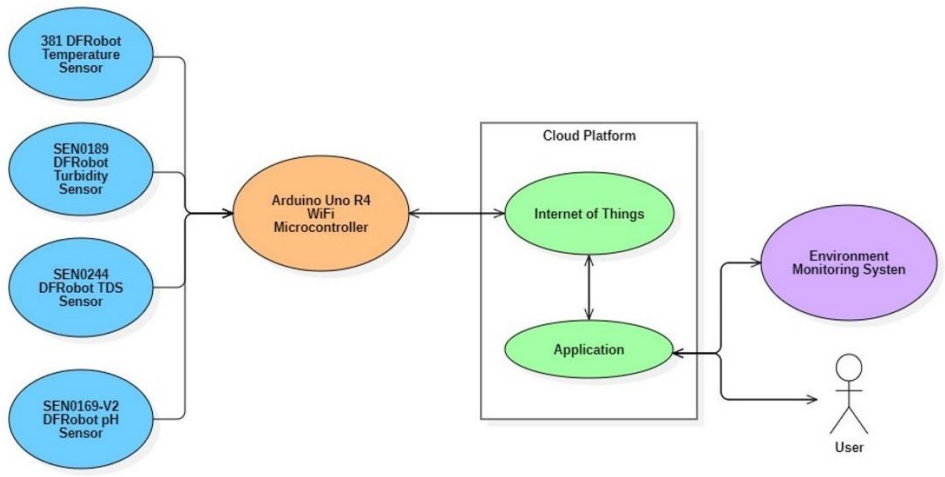


Figure 2: Flowchart of the system design.

Evaluation

The evaluation portion of our project involved testing all our sensor to see if they are reporting accurate readings. Since it is difficult to obtain secondary sensors to verify our readings, we deferred to using reference liquids as our benchmarks. For the testing of the turbidity and dissolved solids sensors (TDS) we used tap water that has a measurement of 1.0 NTU and 300 ppm. The turbidity measures the "cloudiness" of the water which is caused by floating particles in the water, expressed in Nethelometric Turbidity Units (NTU). Our turbidity sensor measured the water to be ~1.1 which is a 10% error from the expected value. The TDS sensor measured the water to be 281 ppm which is a 6.33% error. Even though our project specifications were aiming for an accuracy of within 5% we still deem slightly higher errors as acceptable because we are measuring very clean water. In addition, additional fine tuning and calibration of the sensors might yield more accurate results. For the temperature we measured a liquid that was 16.2℃ (confirmed by secondary sensor). Our temperature sensor measured the temperature to be 16.5℃ which is within the acceptable ± 0.5℃ requirement of our specifications. For the pH sensor, we measured two standard buffer solutions of 7.0pH and 4.0pH. The measurements were 7.3pH and 3.9pH respectively which falls within 5% error of our specification.

The maximum power consumption of our system was also calculated using datasheets. Please note that we have used maximum voltage and current draw for the calculations to account for the worst case scenario:

TDS Sensor:	5V * 6.0mA = 0.03W
Turbidity sensor:	5V * 40mA = 0.2W
Temperature sensor:	5V * 6.0mA = 0.03W
Acidity (pH):	5V * 40mA = 0.2W
Arduino UNO R4 WiFi Board:	5V * 50mA =0.05W
WiFi module:	5V * 150mA = 0.75W
TOTAL:	**1.26W**

Based on this power consumption our chosen battery bank of 38.48Whr will last 30.54 hours which meets our spec that calls for the battery lifetime to be a minimum of 24 hours.

Additional Topics

The inception of this project was strongly motivated by environmental considerations, we specifically focused on how it can contribute to preserving the cleanliness of coastal waters. To determine what constitutes “clean” water, we consulted some of the relevant codes and standards. This was done to ensure that the project implementation aligns with the industry norms. We established some benchmarks for water testing; our test procedures were outlined in the Evaluation section. The Canadian standards for ocean water quality were difficult to find, we have searched Environmental and Workplace Health as well as Statistics Canada for guidelines but unfortunately their guidelines were for fresh and drinking water. Due to the scope and timeline for this project we limited our research for the oceanwater guidelines as the main focus was on making the sensors work and implementing IoT interface. However, we acknowledge that with further development of our prototype, these standards need to be researched and clearly defined. There were several implications that we considered when developing the prototype. Taking into account the social implications, we determined that our project could address a pressing global need for an accessible and affordable solution to monitor water quality. By having a minimal budget, the system becomes more attainable for diverse communities, which includes those in the economically challenged regions. Ethically, the project aims to contribute to solutions addressing environmental challenges. We recognize that water is a vital resource that must be protected for the benefit of all ecosystems on the planet and we must have an approach that aligns with everyone’s interests. A potential ethical challenge that we foresee with IoT technology is the fact that the system involves real-time data transmission to a cloud platform. Since the data might contain some sensitive information it is important to put measures in place to prevent unauthorized access.

Conclusion

In conclusion, our project aims to address the critical need for affordable and real-time water quality monitoring. By integrating Internet of Things (IoT) technology, our system offers a cost-effective solution under $400. The emphasis on simplicity, affordability, and customization allows for deploying multiple specialized monitoring stations, covering extensive areas and facilitating comprehensive data collection.

The literature review delves into three potential solutions for water quality monitoring: buoy-based systems, autonomous surface vehicles (ASVs), and fixed-station shore-based monitoring. Each method presents distinct advantages and challenges, making the choice dependent on specific project requirements. The review highlights the importance of considering cost, location, maintenance needs, and data collection goals when selecting a monitoring system.

Our proposed design incorporates an Arduino UNO R4 WiFi microcontroller and sensors measuring dissolved solids, suspended solids, pH, and temperature. The system aims for simplicity by being housed in a waterproof enclosure with a portable power supply. The chosen IoT interface, Arduino Cloud, enables real-time data visualization through a user-friendly dashboard accessible via a web browser or mobile application.

In the evaluation phase we tested our water quality monitoring system by assessing the accuracy of the readings from the sensors. The turbidity and TDS readings were slightly above the margin of error of 5% but we have deemed them to be acceptable. The temperature and pH readings were both within the margin of error for several different liquids. The power requirements of our system were met and the system can be continuously power for over 24 hours.

The project timeline constraints required the team to focus on completing the system within 42 days from the project inception to the final presentation. To pursue the project further, collaboration and testing would be crucial for refining the design and ensuring the system's effectiveness under different environmental conditions. Our project aims to contribute a practical and accessible solution that aligns with the growing demand for sustainable and efficient water resource management.

References

[1] S. Chandra, A. Mason, "*Smart Sensors for Real-Time Water Quality Monitoring"*, Heidelberg, Germany, Springer, 2013

[2] S. D. Pizzo, A. De Martino, G. De Viti, R. L. Testa and G. De Angelis, "IoT for Buoy Monitoring System," *2018 IEEE International Workshop on Metrology for the Sea; Learning to Measure Sea Health Parameters (MetroSea)*, Bari, Italy, 2018, pp. 232-236

[3] A. T. Demetillo, M. V. Japitana, and E. B. Taboada, "A system for monitoring water quality in a large aquatic area using wireless sensor network technology," *Sustainable environment research*, vol. 29, no. 1, pp. 1–9, 2019

[4] H.-Y. Lu *et al.*, "A Low-Cost AI Buoy System for Monitoring Water Quality at Offshore Aquaculture Cages," *Sensors*, vol. 22, no. 11, p. 4078, May 2022

[5] J. D. Medina *et al.*, "Open-source low-cost design of a buoy for remote water quality monitoring in fish farming," *PloS one*, vol. 17, no. 6, pp. e0270202–e0270202, 2022

[6] J. R. Higinbotham, P. G. Hitchener and J. R. Moisan, "Development of a New Long Duration Solar Powered Autonomous Surface Vehicle," OCEANS 2006, Boston, MA, USA, 2006, pp. 1-6, doi: 10.1109/OCEANS.2006.306874.

[7] J. Moulton et al., "An Autonomous Surface Vehicle for Long Term Operations," OCEANS 2018 MTS/IEEE Charleston, Charleston, SC, USA, 2018, pp. 1-10, doi: 10.1109/OCEANS.2018.8604718.

[8] Su, N., Wang, J.-B., Zeng, C., Zhang, H., Lin, M., & Li, G. Y. (2022). Unmanned Surface Vehicle Aided Maritime Data Collection Using Deep Reinforcement Learning. IEEE Internet of Things Journal, 9(20), 1–1. doi: 10.1109/JIOT.2022.3168589

[9] P. Kimball et al., "The WHOI Jetyak: An autonomous surface vehicle for oceanographic research in shallow or dangerous waters," 2014 IEEE/OES Autonomous Underwater Vehicles (AUV), Oxford, MS, USA, 2014, pp. 1-7, doi: 10.1109/AUV.2014.7054430.

[10] F. Regan et al., "A demonstration of wireless sensing for long term monitoring of water

quality," 2009 IEEE 34th Conference on Local Computer Networks, Zurich, Switzerland, 2009, pp. 819-825, doi: 10.1109/LCN.2009.5355047.

[11] A. Anvari, J. Delos Reyes, E. Esmaeilzadeh, A. Jarvandi, N. Langley and K. R. Navia, "Designing an automated water quality monitoring system for West and Rhode Rivers," 2009 Systems and Information Engineering Design Symposium, Charlottesville, VA, USA, 2009, pp. 131-136, doi: 10.1109/SIEDS.2009.5166167.

ECE 399- Engineering Design Project I
Project Delivery Plan

December 15th, 2023

—

Ocean Water Quality Monitoring System

Group #2

Teaching Assistant
Khalid Al-Hammuri

Members:
Jenna Hilderman
Brandon Ip
Bruce Lynch
Vlad Maslovski

Overview

This document provides you with a fillable template for three main deliverables to help you manage your project efficiently. First, you will have a template for the project charter to define project scope, team, and boundaries. Second, you have to fill in the prioritized features list that you will be using during the design. Third, the project delivery plan, which is the timeline of your project delivery that includes the selected features deliverables in each time interval.

Goals

1. Establish the project management that defines the boundaries and deliverables of your project.
2. Provide the project charter at the kickoff meeting (First meeting with the team).
3. List the project features and prioritize them.
4. Define the project delivery plan in a series of timely milestones.

Important Remarks

This document should be a living object and be updated frequently during the whole project life. Please be aware that the initial plans, including the selected features and delivery plan, might be changed while you progress and have new findings. You may consult with the TA to approve any changes to your plan.

Part 1: Project Charter

Table 1: Project Charter Template

ECE399 Project Charter			
Project Title	Ocean Water Quality Monitoring System		
Project Manager (Team Leader) You may assign a different manager from the team member in each cycle.	Vlad Maslovski	Kick-off Meeting Date (First meeting)	September 20th, 2023
Project Sponsor (TA)	Khalid Al-Hammuri	Project Manager Approval	
Project Summary (Scope)	Our team aims to develop a comprehensive water quality monitoring device. The system will provide real-time data on water conditions, including temperature, pH, turbidity, and conductivity. The device will be environmentally safe.		
Project Goals	**Sensor Choice**: Choose appropriate water sensors to measure temperature, pH, turbidity, and conductivity. **Develop and Implement IoT code**: Develop a data acquisition system to collect the sensor readings that can be monitored in real time. **Alerting System:** Integrate an alerting mechanism to notify users of critical water quality changes or anomalies. **Documentation:** Produce documentation throughout the development of the project.		

ECE399 Project Charter		
Project Assumptions and Constraints. *Time and Budget are for Cost and Benefits Analysis*. *You must define scope boundaries.*	Time	11 Weeks
	Budget	$200-$400
	Scope (Inside)	Develop a comprehensive water quality monitoring device. The system will provide real-time data on water conditions, including temperature, pH, turbidity, and conductivity. The device will be environmentally safe.
	Scope (Outside) Things to avoid	Environmentally destructive materials. Time consuming optional features.
Milestones	Date	Milestone Description
	Oct 19th, 2023	Completed project delivery plan. Begin building prototype.
	Nov 6th, 2023	The prototype is built and successfully collects accurate data from the sensors.
	Nov 15th, 2023	Prototype successfully transmits data.
	Nov 30th, 2023	Prototype successfully transmits data accurately in real time. If time work on an alert system.
	Dec 1th, 2023	Deliver fully developed prototype
Project Risks		Project Main Requirements
1. Water damage 2. Environmental disruption		1. Include waterproofing 2. Research where to implement and potentially

<table>
<tr><th colspan="2">ECE399 Project Charter</th></tr>
<tr><td>3. Inaccuracies</td><td>disruptive chemicals in the device.
3. Choosing an appropriate measuring rate</td></tr>
<tr><td>Stakeholders (Sponsors, supervisors, clients)</td><td>
<table>
<tr><th>Name</th><th>Role</th><th>Responsibility</th></tr>
<tr><td>Khalid Al-Hammuri</td><td>Supervisor</td><td>Approve Project Charter
Evaluate Final Prototype</td></tr>
<tr><td>Ocean Networks Canada</td><td>Client</td><td>Evaluate Final Prototype</td></tr>
<tr><td>Researchers/ Scientists</td><td>Clients</td><td>Potential client</td></tr>
</table>
</td></tr>
<tr><td>Project Team</td><td>
<table>
<tr><th>Name</th><th>Role</th><th>Responsibility</th></tr>
<tr><td>Jenna Hilderman</td><td>Technical Writer
Software/Hardware Developer</td><td>• Project Delivery Plan
• Executive Summary
• Conclusion of literature review
• Write IoT code</td></tr>
<tr><td>Brandon Ip</td><td>Technical Writer
Software/Hardware Developer</td><td>• Write the Introduction and Goal
• Write the Introduction in the literature review.</td></tr>
<tr><td>Bruce Lynch</td><td>Technical Writer
Software/Hardware Developer</td><td>In the literature review write the:
• Autonomous Underwater Vehicles section</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th colspan="4">ECE399 Project Charter</th></tr>
<tr><td></td><td colspan="3"><table>
<tr><td></td><td></td><td>Fixed-Station Monitoring section
Solution Comparison section</td></tr>
<tr><td>Vlad Maslovski</td><td>Project Manager
Technical Writer
Software/Hardware Developer</td><td>PowerPoint presentation
Requirement specifications
Write about the buoy monitoring system
Project timeline (GANTT chart)
Order Parts</td></tr>
</table></td></tr>
<tr><td>Approval (must be approved by TA)</td><td>Title & Name:</td><td colspan="2">_____ Date: _____</td></tr>
</table>

Part 2: Main Prioritized Features the Project.

Table 2: Main Prioritized Features

No.	Feature Name (Task)	Impact (0-5)	Effort (0-5)	Score = Impact + Effort	Feature Dependency	Priority (Numerical)
1	Create a Project Delivery Plan	3	3	6	none	2
2	Reasearch Project	4	5	9	none	1
3	Write Project Research Document	2	5	7	2	3
4	Design Protoptype	5	4	9	2, 3	4
5	Research components	4	3	7	2	5
6	Purchase components	5	1	6	2, 5	6
7	Build Prototype	3	2	5	2, 5, 6	7
8	Outline code	3	3	6	2, 5	8
9	Write IoT code	5	5	10	2, 5, 8	9
10	Implement IoT Code	5	3	8	2, 5, 8, 9	10

Part 3: Roadmap of your Features Delivery Plan

Project Milestones	Status	Related files	Notes
First Meeting and Project Charter	Completed	Current document	Initial Project
Define Project Scope and Features	Completed	Current document	Select the appropriate features
Launch Project (Implementation Phase)	Completed	Parts ordered	Implement your project physically
Project Review and Performance Analysis	Completed	Budget and Project Timeline	Your TA and Project Team should Approve the Project
Project Close	Completed	Main Documentation	Prepare the project's Final demo.

Note: Please see detailed project timeline on the next page.

PROJECT TIMELINE

PROJECT TITLE	Ocean Water Quality Monitoring System	**Group #**	2
PROJECT MANAGER	Vlad Maslovski	**DATE**	12-15-23

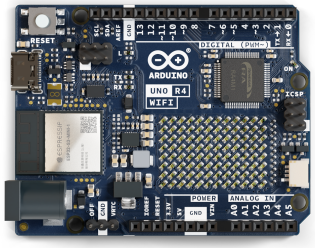
This timeline provides you with a guideline on how to create a timeline for your project. You have different phases and details suggestions for each phase for the tasks or the

Color Code
In progress
Completed
Not comple

PHASE		DETAILS SUGGESTIONS	Q1				Q2				Q3		
			Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11
	PROJECT WEEK:	Enter the date of the first Monday of each m	2nd Oct		19			6th Nov				5h Dec	Dec. 11
1	**Project Conception and Initiation**	- Project Charter	Project Charter		Finalize Charter								
		- Plan Review	Review Plan	Complete Initial Timeline									
		- Initiation		Order M/C and sensors		Order Remaining Parts							
2	**Project Definition and Planning (Define Features)**	- Scope and Define Features	Define Scope	Define Features	Finalize Features								
		- Budget	Select Microcontroller	Select Sensors	Finalize Budget								
		- Work Breakdown Schedule	Documentation Schedule			First Demo Preparation Schedule			Final Demo Schedule				
		- Define Team	Define Team Documentation Responsiblities		Define Project Roles								
		- Communication Plan	Finalize Communication Plan										
		- Risk Management	Risk Management Plan	Finalize risk management Plan			Refine Risk Management Plan						
3	**Project Launch & Execution**	- Status and Tracking				1st Demo Prep	Ready for 1st Demo		Final Demo Prep	Ready for Final Demo			
		- Features Implementaion			Wire Sensors to Arduino	C code to read data from sensors		C code to process data	Internet of Things integration and Interface				
		- KPIs (Project Evaluation by TA)				List of KPIs before 1st Demo			Finaze KPI list before 1st Demo				
4	**Project Performance & Control**	- Forecasts (Can You Finish Project as Planned)				Evaluate readiness before 1st Demo			Evaluate readiness before Final Demo				
		- Effort and Cost Tracking		Budget spreadsheet	Effort Spreadsheet	Updates to Effort and Budget Spreadsheet			Finalize effort and Budget				
		- Performance			Identifiy potential challenges		1st Demo: Quality Control			Final Demo: Quality Control	Evaluate Final Performance		
5	**Project Close**	- List what features works and what does not							List what features work and what does not	Explain reasons for setbacks			
		- Final Report Submission and Demo	Initial Documentation Submission				Updates to Documentation				Final Demo	Finalize Documentation	Documentation Submission

PROJECT END

Product Reference Manual
SKU: ABX00087



Description

The Arduino® UNO R4 WiFi is the first UNO board to feature a 32-bit microcontroller and an ESP32-S3 Wi-Fi® module (ESP32-S3-MINI-1-N8). It features a RA4M1 series microcontroller from Renesas (R7FA4M1AB3CFM#AA0), based on a 48 MHz Arm® Cortex®-M4 microprocessor. The UNO R4 WiFi's memory is larger than its predecessors, with 256 kB flash, 32 kB SRAM and 8 kB of EEPROM.

The RA4M1's operating voltage is fixed at 5 V, whereas the ESP32-S3 module is 3.3 V. Communication between these two MCUs is performed via a logic-level translator (TXB0108DQSR).

Target areas:

Maker, beginner, education

Features

The **R7FA4M1AB3CFM#AA0**, often referred to as RA4M1 in this datasheet, is the main MCU on the UNO R4 WiFi, connected to all pin headers on the board as well as all communication buses.

- **Overview**
  - 48 MHz Arm® Cortex®-M4 microprocessor with a floating point unit (FPU)
  - 5 V operating voltage
  - Real-time Clock (RTC)
  - Memory Protection Unit (MPU)
  - Digital-to-analog Converter (DAC)
- **Memory**
  - 256 kB Flash Memory
  - 32 kB SRAM
  - 8 kB Data Memory (EEPROM)
- **Peripherals**
  - Capacitive Touch Sensing Unit (CTSU)
  - USB 2.0 Full-Speed Module (USBFS)
  - 14-bit ADC
  - Up to 12-bit DAC
  - Operational Amplifier (OPAMP)
- **Power**
  - Operating voltage for RA4M1 is 5 V
  - Recommended input voltage (VIN) is 6-24 V
  - Barrel jack connected to VIN pin (6-24 V)
  - Power via USB-C® at 5 V
- **Communication**
  - 1x UART (pin D0, D1)
  - 1x SPI (pin D10-D13, ICSP header)
  - 1x I2C (pin A4, A5, SDA, SCL)
  - 1x CAN (pin D4, D5, external transceiver is required)

See the full datasheet for the R7FA4M1AB3CFM#AA0 in the link below:

- R7FA4M1AB3CFM#AA0 datasheet

The **ESP32-S3-MINI-1-N8** is the secondary MCU with a built-in antenna for Wi-Fi® & Bluetooth® connectivity. This module operates on 3.3 V and communicates with the RA4M1 using a logic-level translator (TXB0108DQSR).

- **Overview**
  - Xtensa® dual-core 32-bit LX7 microprocessor
  - 3.3 V operating voltage
  - 40 MHz crystal oscillator

- **Wi-Fi®**
  - Wi-Fi® support with 802.11 b/g/n standard (Wi-Fi® 4)
  - Bit rate at up to 150 Mbps
  - 2.4 GHz band
- **Bluetooth®**
  - Bluetooth® 5

See the full datasheet for the ESP32-S3-MINI-1-N8 in the link below:

- ESP32-S3-MINI-1-N8 datasheet

CONTENTS

1 The Board

1.1 Application Examples

The UNO R4 WiFi is part of the first UNO series of 32-bit development boards, being previously based on 8-bit AVR microcontrollers. There are thousands of guides, tutorials and books written about the UNO board, where the UNO R4 WiFi continues its legacy.

The board features 14 digital I/O ports, 6 analog channels, dedicated pins for I2C, SPI and UART connections. It has a significantly larger memory: 8 times more flash memory (256 kB) and 16 times more SRAM (32 kB). With a 48 MHz clock speed, it is also 3x faster than its predecessors.

In addition, it features an ESP32-S3 module for Wi-Fi® & Bluetooth® connectivity, as well as a built-in 12x8 LED matrix, making it one of the most visually unique Arduino boards to date. The LED matrix is fully programmable, where you can load anything from still frames to custom animations.

Entry-level projects: If this is your first project within coding and electronics, the UNO R4 WiFi is a good fit. It is easy to get started with, and it has a lot of online documentation.

Easy IoT applications: build projects without writing any networking code in the Arduino IoT Cloud. Monitor your board, connect it with other boards and services, and develop cool IoT projects.

LED Matrix: the 12x8 LED matrix on the board can be used for showing animations, text scrolling, create mini-games and much more, being the perfect feature to give your project more personality.

1.2 Related Products

- UNO R3
- UNO R3 SMD
- UNO R4 Minima

Rating

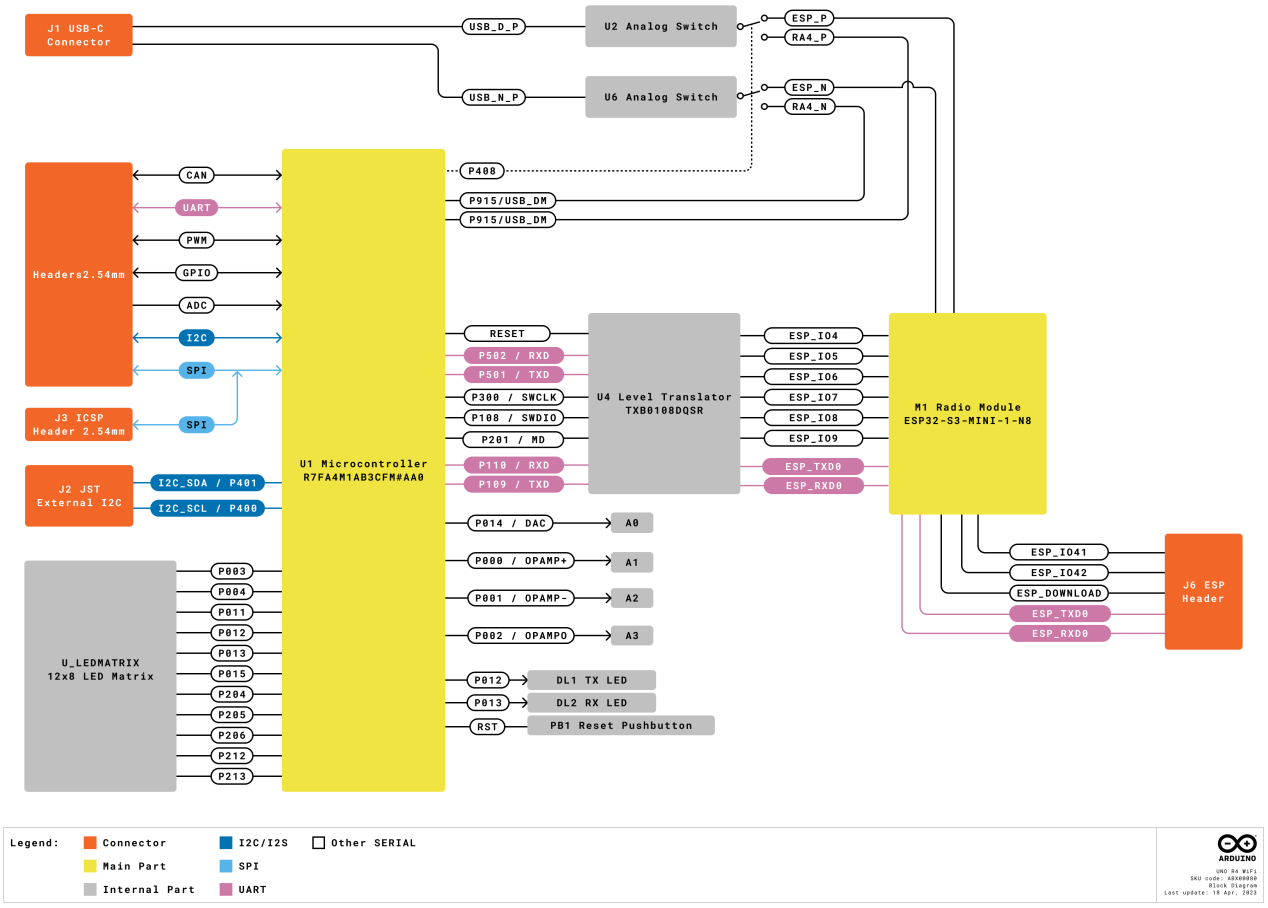
2 Recommended Operating Conditions

Symbol	Description	Min	Typ	Max	Unit
V_{IN}	Input voltage from VIN pad / DC Jack	6	7.0	24	V
V_{USB}	Input voltage from USB connector	4.8	5.0	5.5	V
T_{OP}	Operating Temperature	-40	25	85	°C

Note: V_{DD} controls the logic level and is connected to the 5V power rail. V_{AREF} is for the analog logic.

Functional Overview

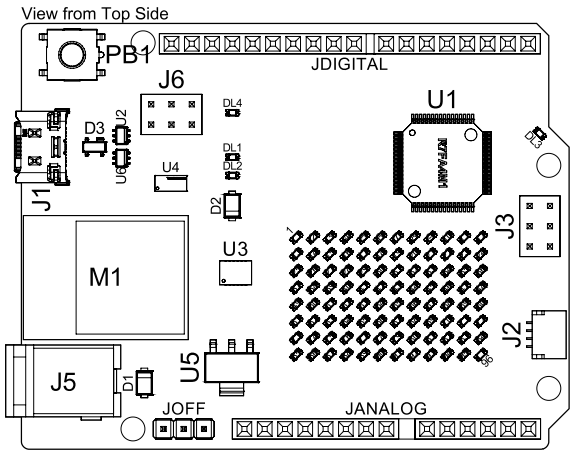
3 Block Diagram



Arduino R4 WiFi Block Diagram

4 Board Topology

4.1 Front View



Top View of Arduino UNO R4 WiFi

Ref.	Description
U1	R7FA4M1AB3CFM#AA0 Microcontroller IC
U2	NLASB3157DFT2G Multiplexer
U3	ISL854102FRZ-T Buck Converter
U4	TXB0108DQSR logic level translator (5 V - 3.3 V)
U5	SGM2205-3.3XKC3G/TR 3.3 V linear regulator
U6	NLASB3157DFT2G Multiplexer
U_LEDMATRIX	12x8 LED Red Matrix
M1	ESP32-S3-MINI-1-N8
PB1	RESET Button
JANALOG	Analog input/output headers
JDIGITAL	Digital input/output headers
JOFF	OFF, VRTC header
J1	CX90B-16P USB-C® connector
J2	SM04B-SRSS-TB(LF)(SN) I2C connector
J3	ICSP header (SPI)
J5	DC Jack
J6	ESP header
DL1	LED TX (serial transmit)

Ref.	Description
DL2	LED RX (serial receive)
DL3	LED Power (green)
DL4	LED SCK (serial clock)
D1	PMEG6020AELRX Schottky Diode
D2	PMEG6020AELRX Schottky Diode
D3	PRTR5V0U2X,215 ESD Protection

5 Microcontroller (R7FA4M1AB3CFM#AA0)

The UNO R4 WiFi is based on the 32-bit RA4M1 series microcontroller, **R7FA4M1AB3CFM#AA0**, from Renesas, which uses a 48 MHz Arm® Cortex®-M4 microprocessor with a floating point unit (FPU).

The operating voltage for the RA4M1 is fixed at 5 V as to be hardware compatible with shields, accessories & circuits based on previous Arduino UNO boards.

The R7FA4M1AB3CFM#AA0 features:

- 256 kB flash / 32 kB SRAM / 8 kB data flash (EEPROM)
- Real-time Clock (RTC)
- 4x Direct Memory Access Controller (DMAC)
- 14-bit ADC
- Up to 12-bit DAC
- OPAMP
- CAN bus

For more technical details on this microcontroller, visit the Renesas - RA4M1 series official documentation.

6 Wi-Fi® / Bluetooth® Module (ESP32-S3-MINI-1-N8)

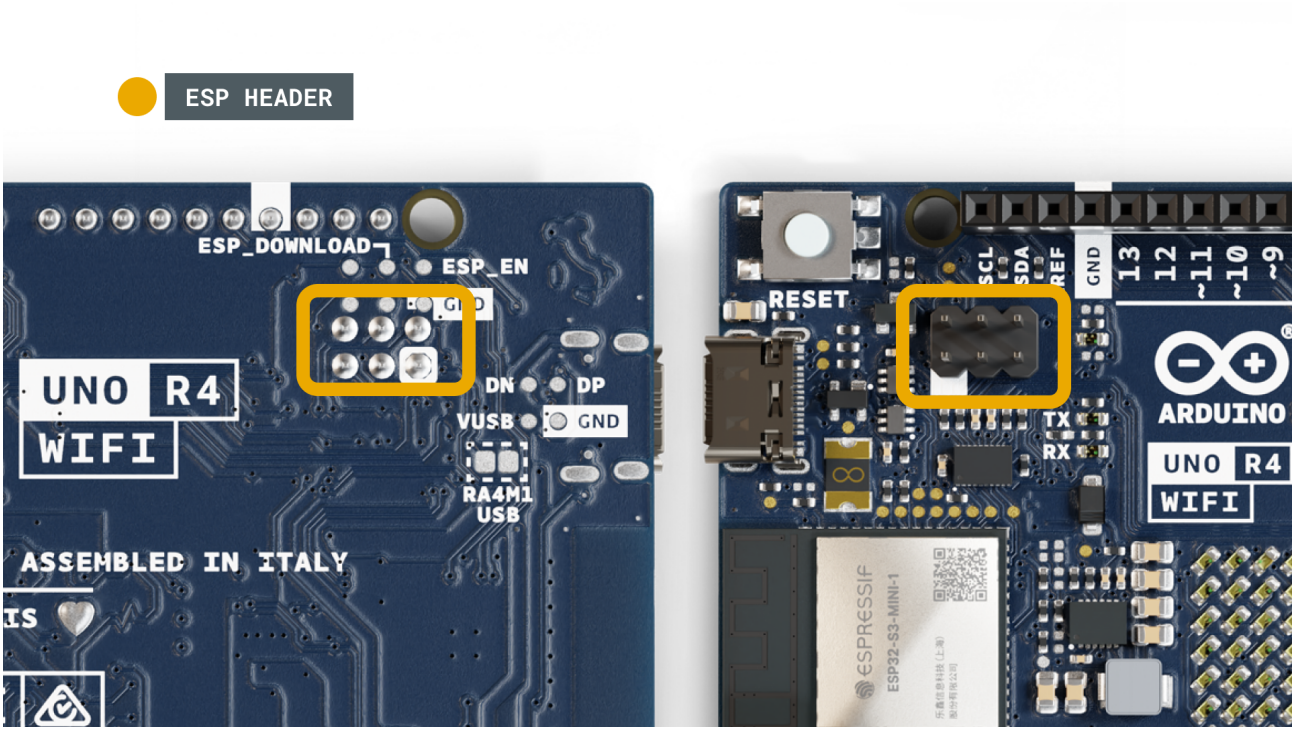
The Wi-Fi® / Bluetooth® LE module on the UNO R4 WiFi is from the ESP32-S3 SoCs. It features the Xtensa® dual-core 32-bit LX7 MCU, a built-in antenna and support for 2.4 GHz bands.

The ESP32-S3-MINI-1-N8 features:

- Wi-Fi® 4 - 2.4 GHz band
- Bluetooth® 5 LE support
- 3.3 V operating voltage
- 384 kB ROM
- 512 kB SRAM
- Up to 150 Mbps bit rate

This module acts as a secondary MCU on the UNO R4 WiFi, and communicates with the RA4M1 MCU using a logic-level translator. Note that this module operates on 3.3 V as opposed to the RA4M1's 5 V operating voltage.

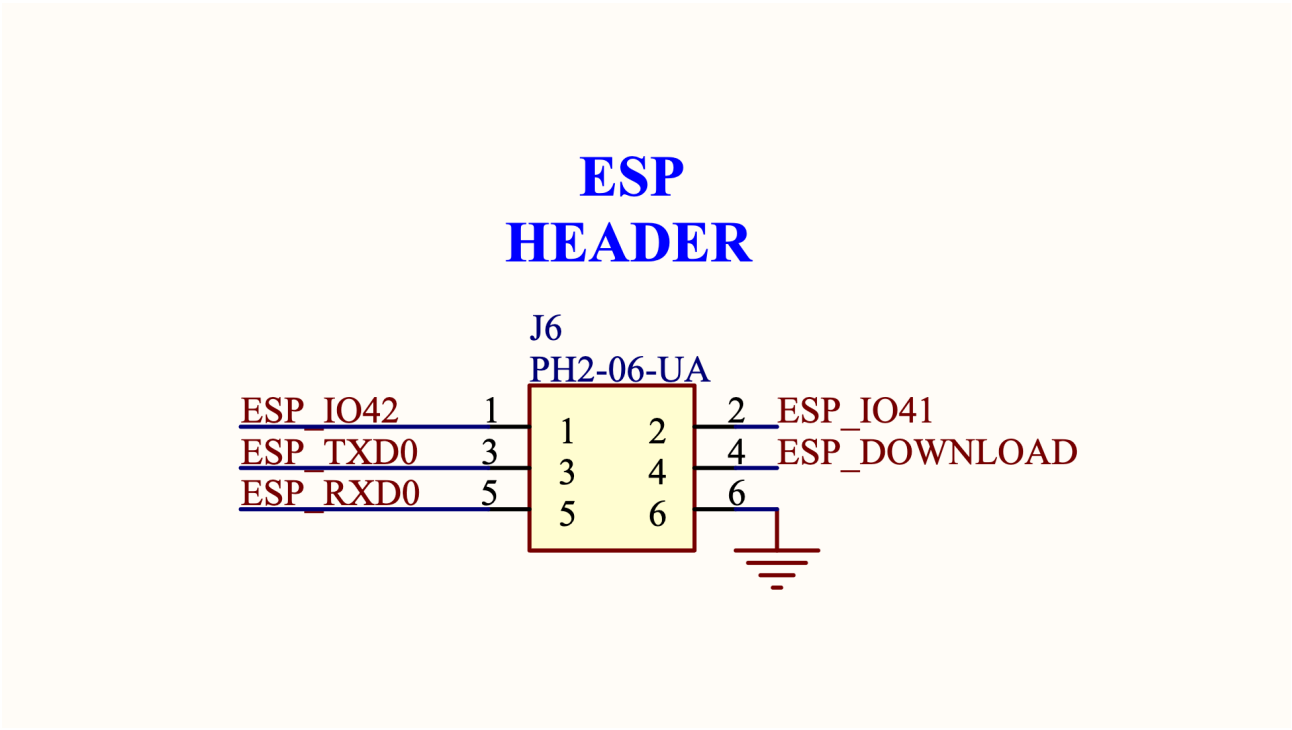
6.1 ESP Header



ESP header.

The header located close to the RESET button can be used to access the ESP32-S3 module directly. The pins accessible are:

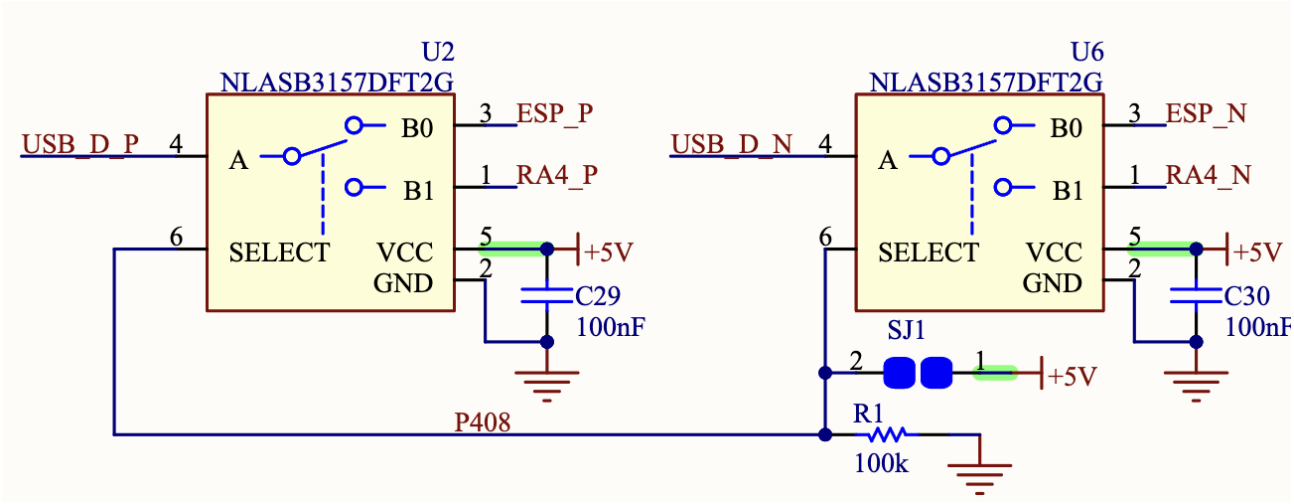
- `ESP_IO42` - MTMS debugging (Pin 1)
- `ESP_IO41` - MTDI debugging (Pin 2)
- `ESP_TXD0` - Serial Transmit (UART) (Pin 3)
- `ESP_DOWNLOAD` - boot (Pin 4)
- `ESP_RXD0` - Serial Receive (UART) (Pin 5)
- `GND` - ground (Pin 6)



ESP header (schematic)

6.2 USB Bridge

When programming the UNO R4 WiFi, the RA4M1 MCU is programmed via the ESP32-S3 module by default. The **U2** and **U6** switches can switch the USB communication to go directly to the RA4M1 MCU, by writing a high state to the P408 pin (D40).



Soldering together the **SJ1** pads permanently sets the USB communication directly to the RA4M1, bypassing the ESP32-S3.

7 USB Connector

The UNO R4 WiFi has one USB-C® port, used to power and program your board as well as sending & receiving serial communication.

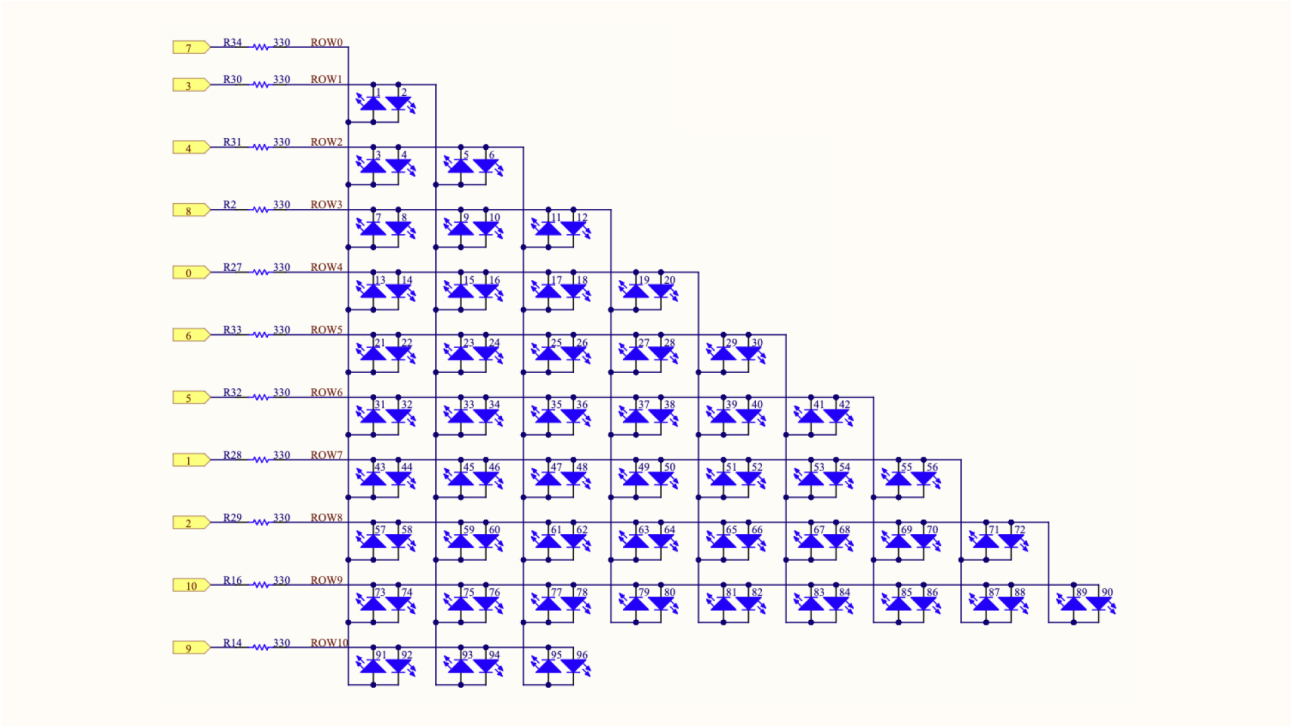
Note: The board should not be powered with more than 5 V via the USB-C® port.

8 LED Matrix

The UNO R4 WiFi features a 12x8 matrix of red LEDs (**U_LEDMATRIX**), connected using the technique known as charlieplexing.

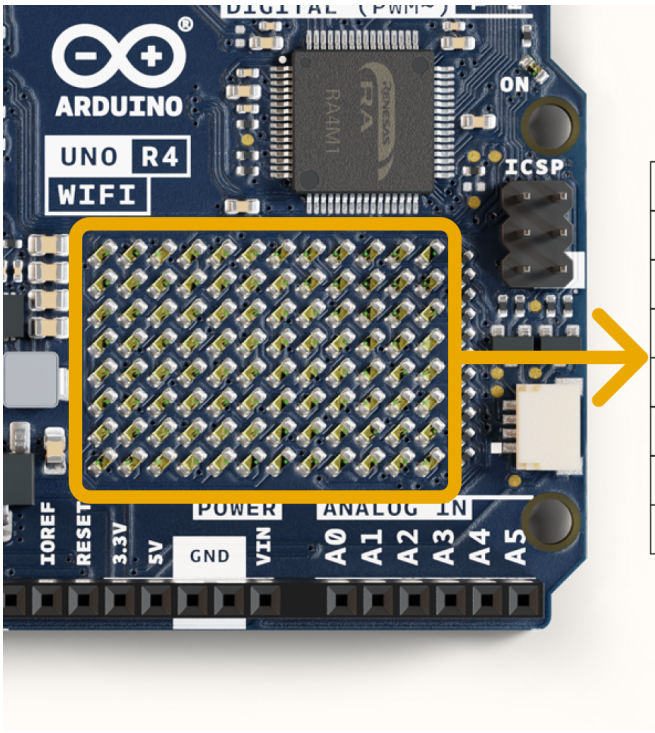
The following pins on the RA4M1 MCU are used for the matrix:

- P003
- P004
- P011
- P012
- P013
- P015
- P204
- P205
- P206
- P212
- P213



LED matrix schematics.

These LEDs can be accessed as an array, using a specific library. See the mapping below:



1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80	81	82	83	84
85	86	87	88	89	90	91	92	93	94	95	96

LED matrix number mapping.

This matrix can be used for a number of projects and prototyping purposes, and supports animation, simple game designs and scrolling text among other things.

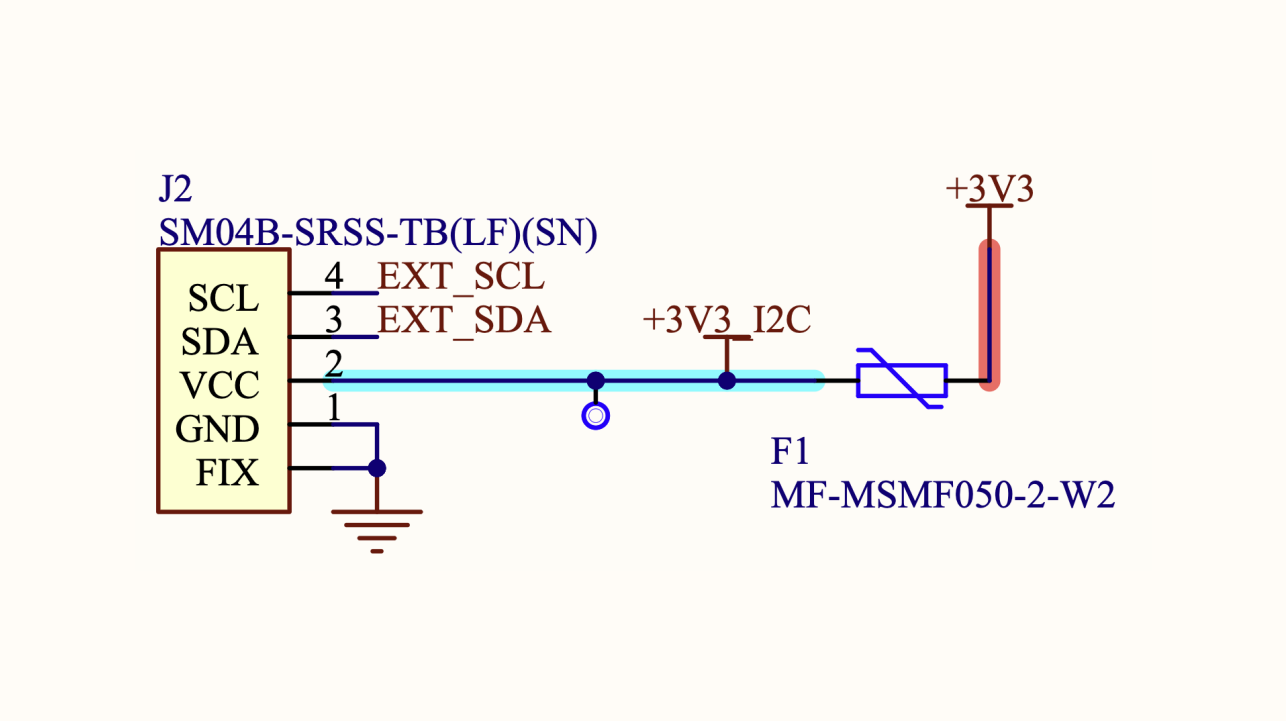
9 Digital Analog Converter (DAC)

The UNO R4 WiFi has a DAC with up to 12-bit resolution attached to the A0 analog pin. A DAC is used to convert a digital signal to an analog signal.

The DAC can be used for signal generation for e.g. audio applications, like generating and altering sawtooth waves.

10 I2C Connector

The I2C connector SM04B-SRSS-TB(LF)(SN) is connected to a secondary I2C bus on the board. Note that this connector is powered via 3.3 V.



I2C connector.

This connector also shares the following pin connections:

JANALOG header

- A4
- A5

JDIGITAL header

- SDA
- SCL

Note: as A4/A5 is connected to the main I2C bus, these should not be used as ADC inputs whenever the bus is in use. You can however connect I2C devices to each of these pins and connectors simultaneously.

11 Power Options

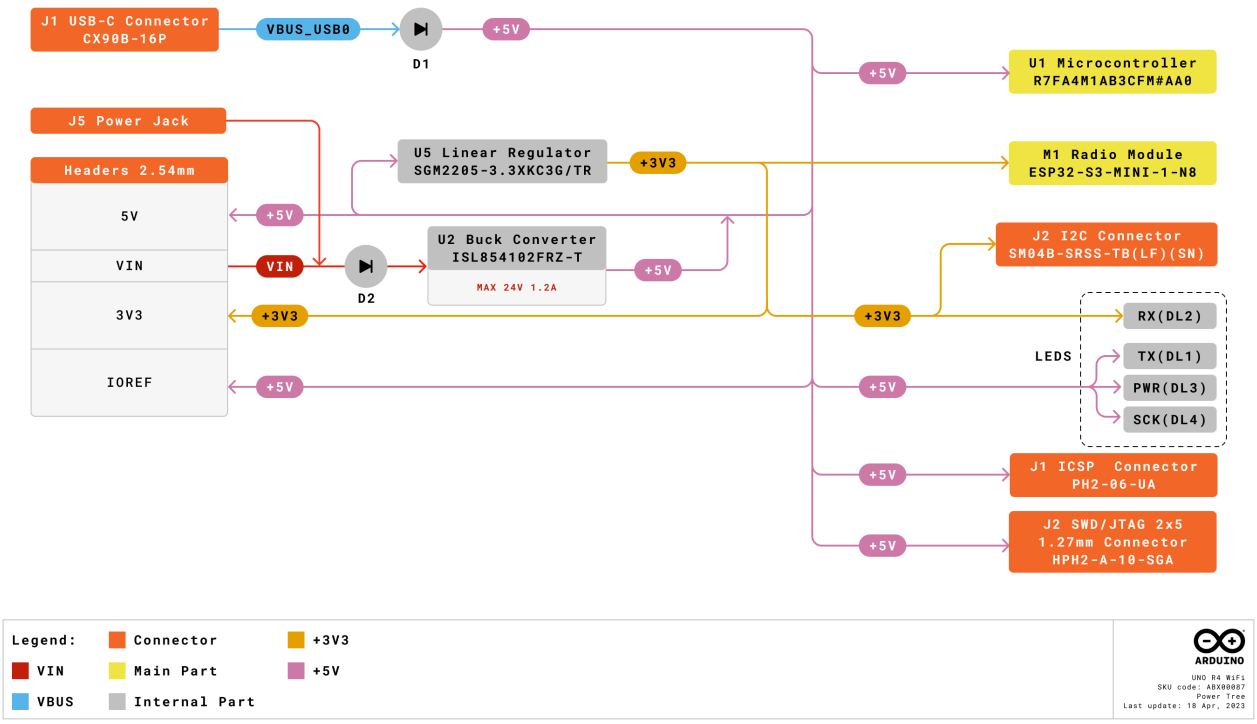
Power can either be supplied via the VIN pin, or via USB-C® connector. If power is supplied via VIN, the ISL854102FRZ buck converter steps the voltage down to 5 V.

Both VUSB and VIN pins are connected to the ISL854102FRZ buck converter, with Schottky diodes in place for reverse polarity & overvoltage protection respectively.

Power via USB supplies about ~4.7 V (due to Schottky drop) to the RA4M1 MCU.

The linear regulator (SGM2205-3.3XKC3G/TR) converts 5 V from either the buck converter or USB, and provides 3.3 V to a number of components, including the ESP32-S3 module.

11.1 Power Tree



Arduino UNO R4 WiFi power tree.

11.2 Pin Voltage

The general operating voltage for UNO R4 WiFi is 5 V, however the ESP32-S3 module's operating voltage is 3.3 V.

Note: It is **very** important that ESP32-S3's pins (3.3 V) do not come in contact with any of the RA4M1's pins (5 V), as this may damage the circuits.

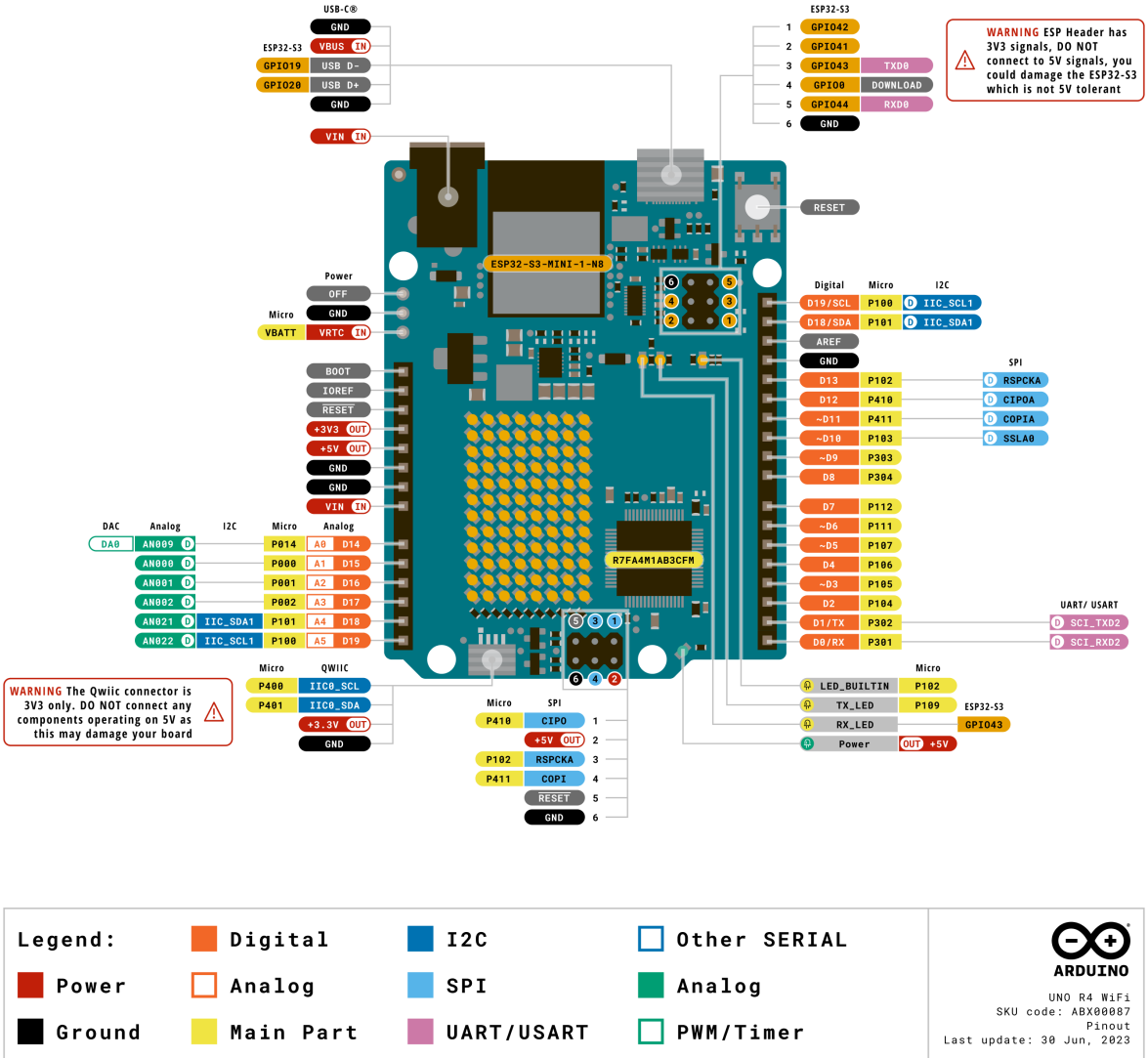
11.3 Pin Current

The GPIOs on the R7FA4M1AB3CFM#AA0 microcontroller can safely handle up to 8 mA of current. Never connect devices that draw higher current directly to a GPIO as this may damage the circuit.

For powering e.g. servo motors, always use an external power supply.

Mechanical Information

12 Pinout



Pinout for UNO R4 WiFi.

12.1 Analog

Pin	Function	Type	Description
1	BOOT	NC	Not Connected
2	IOREF	IOREF	Reference for digital logic V - connected to 5 V
3	Reset	Reset	Reset
4	+3V3	Power	+3V3 Power Rail
5	+5V	Power	+5V Power Rail
6	GND	Power	Ground
7	GND	Power	Ground
8	VIN	Power	Voltage Input
9	A0	Analog	Analog input 0 / DAC
10	A1	Analog	Analog input 1 / OPAMP+
11	A2	Analog	Analog input 2 / OPAMP-
12	A3	Analog	Analog input 3 / OPAMPOut
13	A4	Analog	Analog input 4 / I2C Serial Datal (SDA)
14	A5	Analog	Analog input 5 / I2C Serial Clock (SCL)

12.2 Digital

Pin	Function	Type	Description
1	SCL	Digital	I2C Serial Clock (SCL)
2	SDA	Digital	I2C Serial Datal (SDA)
3	AREF	Digital	Analog Reference Voltage
4	GND	Power	Ground
5	D13/SCK/CANRX0	Digital	GPIO 13 / SPI Clock / CAN Receiver (RX)
6	D12/CIPO	Digital	GPIO 12 / SPI Controller In Peripheral Out
7	D11/COPI	Digital	GPIO 11 (PWM) / SPI Controller Out Peripheral In
8	D10/CS/CANTX0	Digital	GPIO 10 (PWM) / SPI Chip Select / CAN Transmitter (TX)
9	D9	Digital	GPIO 9 (PWM~)
10	D8	Digital	GPIO 8
11	D7	Digital	GPIO 7
12	D6	Digital	GPIO 6 (PWM~)
13	D5	Digital	GPIO 5 (PWM~)
14	D4	Digital	GPIO 4
15	D3	Digital	GPIO 3 (PWM~) / Interrupt Pin
16	D2	Digital	GPIO 2 / Interrupt Pin
17	D1/TX0	Digital	GPIO 1 / Serial 0 Transmitter (TX)
18	D0/TX0	Digital	GPIO 0 / Serial 0 Receiver (RX)

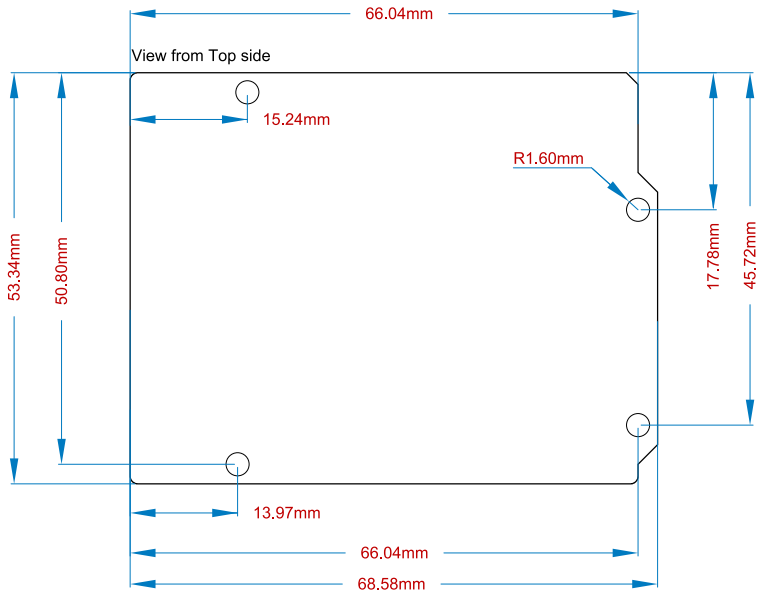
12.3 OFF

Pin	Function	Type	Description
1	OFF	Power	For controlling power supply
2	GND	Power	Ground
1	VRTC	Power	Battery connection to power RTC only

12.4 ICSP

Pin	Function	Type	Description
1	CIPO	Internal	Controller In Peripheral Out
2	+5V	Internal	Power Supply of 5 V
3	SCK	Internal	Serial Clock
4	COPI	Internal	Controller Out Peripheral In
5	RESET	Internal	Reset
6	GND	Internal	Ground

13 Mounting Holes And Board Outline



Top side Mechanical View of Arduino UNO R4 WiFi

14 Board Operation

14.1 Getting Started - IDE

If you want to program your UNO R4 WiFi while offline you need to install the Arduino® Desktop IDE **[1]**. To connect the UNO R4 WiFi to your computer, you will need a Type-C® USB cable, which can also provide power to the board, as indicated by the LED (DL1).

14.2 Getting Started - Arduino Web Editor

All Arduino boards, including this one, work out-of-the-box on the Arduino® Web Editor **[2]**, by just installing a simple plugin.

The Arduino Web Editor is hosted online, therefore it will always be up-to-date with the latest features and support for all boards. Follow **[3]** to start coding on the browser and upload your sketches onto your board.

14.3 Getting Started - Arduino IoT Cloud

All Arduino IoT enabled products are supported on Arduino IoT Cloud which allows you to log, graph and analyze sensor data, trigger events, and automate your home or business.

14.4 Online Resources

Now that you have gone through the basics of what you can do with the board you can explore the endless possibilities it provides by checking existing projects on Arduino Project Hub **[4]**, the Arduino Library Reference **[5]**, and the online store **[6]**; where you will be able to complement your board with sensors, actuators and more.

14.5 Board Recovery

All Arduino boards have a built-in bootloader which allows flashing the board via USB. In case a sketch locks up the processor and the board is not reachable anymore via USB, it is possible to enter bootloader mode by double-tapping the reset button right after the power-up.

Certifications

15 Declaration of Conformity CE DoC (EU)

We declare under our sole responsibility that the products above are in conformity with the essential requirements of the following EU Directives and therefore qualify for free movement within markets comprising the European Union (EU) and European Economic Area (EEA).

16 Declaration of Conformity to EU RoHS & REACH 211 01/19/2021

Arduino boards are in compliance with RoHS 2 Directive 2011/65/EU of the European Parliament and RoHS 3 Directive 2015/863/EU of the Council of 4 June 2015 on the restriction of the use of certain hazardous substances in electrical and electronic equipment.

Substance	Maximum Limit (ppm)
Lead (Pb)	1000
Cadmium (Cd)	100
Mercury (Hg)	1000
Hexavalent Chromium (Cr6+)	1000
Poly Brominated Biphenyls (PBB)	1000
Poly Brominated Diphenyl ethers (PBDE)	1000
Bis(2-Ethylhexyl} phthalate (DEHP)	1000
Benzyl butyl phthalate (BBP)	1000
Dibutyl phthalate (DBP)	1000
Diisobutyl phthalate (DIBP)	1000

Exemptions : No exemptions are claimed.

Arduino Boards are fully compliant with the related requirements of European Union Regulation (EC) 1907 /2006 concerning the Registration, Evaluation, Authorization and Restriction of Chemicals (REACH). We declare none of the SVHCs (https://echa.europa.eu/web/guest/candidate-list-table), the Candidate List of Substances of Very High Concern for authorization currently released by ECHA, is present in all products (and also package) in quantities totaling in a concentration equal or above 0.1%. To the best of our knowledge, we also declare that our products do not contain any of the substances listed on the "Authorization List" (Annex XIV of the REACH regulations) and Substances of Very High Concern (SVHC) in any significant amounts as specified by the Annex XVII of Candidate list published by ECHA (European Chemical Agency) 1907 /2006/EC.

17 Conflict Minerals Declaration

As a global supplier of electronic and electrical components, Arduino is aware of our obligations with regards to laws and regulations regarding Conflict Minerals, specifically the Dodd-Frank Wall Street Reform and Consumer Protection Act, Section 1502. Arduino does not directly source or process conflict minerals such as Tin, Tantalum, Tungsten, or Gold. Conflict minerals are contained in our products in the form of solder, or as a component in metal alloys. As part of our reasonable due diligence Arduino has contacted component suppliers within our supply chain to verify their continued compliance with the regulations. Based on the information received thus far we declare that our products contain Conflict Minerals sourced from conflict-free areas.

18 FCC Caution

Any Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions:

(1) This device may not cause harmful interference

(2) this device must accept any interference received, including interference that may cause undesired operation.

FCC RF Radiation Exposure Statement:

1. This Transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.
2. This equipment complies with RF radiation exposure limits set forth for an uncontrolled environment.
3. This equipment should be installed and operated with a minimum distance of 20 cm between the radiator & your body.

Note: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

English: User manuals for licence-exempt radio apparatus shall contain the following or equivalent notice in a conspicuous location in the user manual or alternatively on the device or both. This device complies with Industry Canada licence-exempt RSS standard(s). Operation is subject to the following two conditions:

(1) this device may not cause interference

(2) this device must accept any interference, including interference that may cause undesired operation of the device.

French: Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes :

(1) l' appareil nedoit pas produire de brouillage

(2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

IC SAR Warning:

English This equipment should be installed and operated with a minimum distance of 20 cm between the radiator and your body.

French: Lors de l' installation et de l' exploitation de ce dispositif, la distance entre le radiateur et le corps est d 'au moins 20 cm.

Important: The operating temperature of the EUT can't exceed 85 °C and shouldn't be lower than -40 °C.

Hereby, Arduino S.r.l. declares that this product is in compliance with essential requirements and other relevant provisions of Directive 2014/53/EU. This product is allowed to be used in all EU member states.

19 Company Information

Company name	Arduino SRL
Company Address	Via Andrea Appiani, 25 - 20900 MONZA□Italy)

20 Reference Documentation

Ref	Link
Arduino IDE (Desktop)	https://www.arduino.cc/en/Main/Software
Arduino IDE (Cloud)	https://create.arduino.cc/editor
Cloud IDE Getting Started	https://docs.arduino.cc/cloud/web-editor/tutorials/getting-started/getting-started-web-editor
Project Hub	https://create.arduino.cc/projecthub?by=part&part_id=11332&sort=trending
Library Reference	https://github.com/arduino-libraries/
Online Store	https://store.arduino.cc/

21 Change Log

Date	Revision	Changes
19/09/2023	4	Update FCC section
25/07/2023	3	Update Pin Table
30/06/2023	2	Update Pinout File
08/06/2023	1	First Release

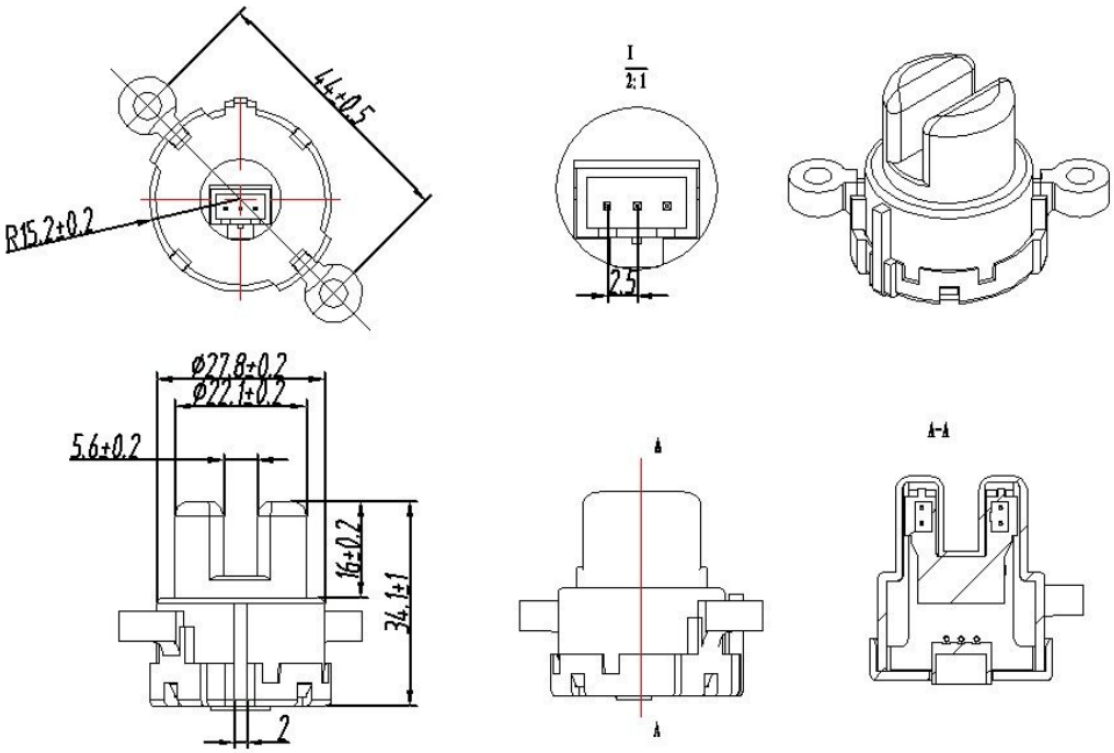
Turbidity sensor

1. Manufacturer Brand: DFRobot

2. Place of Origin: China

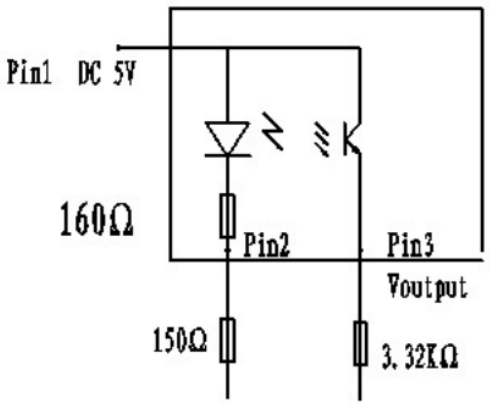
3. Ordering Code: M021.00084

4. Outline Dimensions:

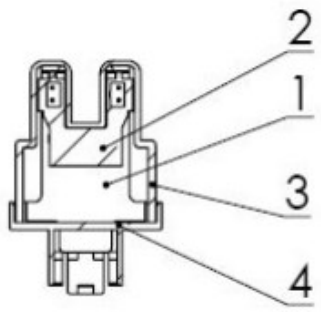


5. Internal Structure:

The sensor and its test circuit



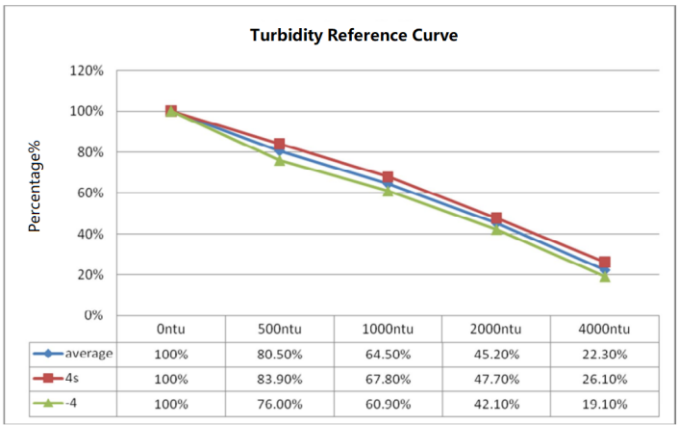
6. The Brand, Parameters And Composition of the Key Components:



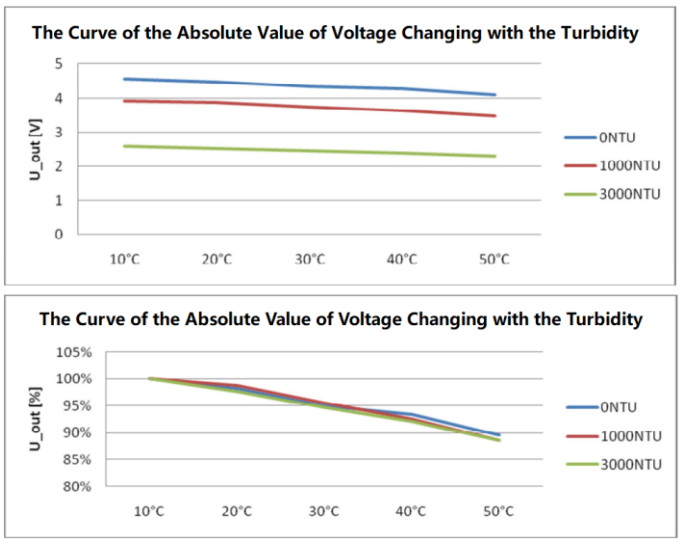
No.	Name	Material Composition	Environmental Standards
1	PCB Components	CM-3, photosensitive element	RoHS
2	Support	PA6+15%	RoHS
3	Shell	PP	RoHS
4	Back Cover	PA6+15%	RoHS

7. Electric Performance Parameters

- Rated Voltage: DC 5 V
- Rated Current: 30mA
- Withstand Voltage: No flash-over or breakdown when applying AC 1500V, 50Hz voltage between the energized terminal and shell for 1S.
- Leakage Current: Leakage between coil and shell < 0.25mA
- Insulation Resistance: The insulation resistance should be greater than 100MΩ when the voltage of DC 500V is applied between the charged spot and the exposed non-charged metal and non-metal.
- Turbidity Reference Curve:



- The Curve of the Absolute Value of Voltage Changing with the Turbidity:



8. Mechanical Properties and Environmental Resistance

- Appearance: No bending of terminals, no cracking of shell, clear and readable printing
- Operating Liquid: Clear water
- Operating Temperature: 5 ℃ ~ 90 ℃
- Storage Temperature: -10 ℃ ~ 90 ℃
- Low Temperature Performance: -20 ℃ (24 Hrs), output deviation is less than 10%
- Thermal Shock Resistance: - 20°C (30min), 85°C (30min), 100 cycles, no visible damage to the product and the output deviation is less than 10%
- Damp Heat Test: After storing the sensor in a damp heat oven with a relative humidity of 85% and temperature of 85 ℃ for 90 hours, its output deviation is less than 10%.
- Salt Spray Test: After placing the sensor in the salt spray box for 168 hours, there is no rust that could cause malfunctions.
- Vibration-resistance Performance: When placed on a vibrating table with an amplitude of 3 mm and a frequency of 22 Hz, after vibration test for 10 minutes in each direction: forward/backward, left/right, and up/down, the sensor keeps intact, no structural deformation, and its output deviation is less than 10%.
- Drop Test: After being dropped from a height of 80 cm from the cement floor, there should be no abnormalities in the sensor such as cracks and deformations in all parts, still holding the ability to meet the requirements of insulation performance.

9. Manufacturer’ s Key Process Parameters

Sensor Test: Put the sensor in clean water with NTU<0.5 for testing, its voltage output is V=4.1±0.3V

10. RoHS Requirements:

This product is RoHS compliant.

11. Packaging Method:

50 pcs/pack
10 packs/carton

12. Precautions for Use:

Operating conditions

- Operating Temperature: 5 °C ~ 90 °C
- Storage Temperature: -10 °C ~ 90 °C

Gravity: Analog TDS Sensor / Meter For Arduino

SKU: SEN0244

TDS (Total Dissolved Solids) indicates that how many milligrams of soluble solids dissolved in one liter of water. In general, the higher the TDS value, the more soluble solids dissolved in water, and the less clean the water is. Therefore, the TDS value can be used as one of the references for reflecting the cleanliness of water.

TDS pen is a widely used equipment to measure TDS value. The price is affordable, and it is easy to use, but it is not able to transmit data to the control system for online monitoring to do some water quality analysis. The professional instrument has high accuracy and can send data to the control system, but the price is expensive for the ordinary people. To this end, we have launched an analog TDS sensor kit which is compatible with Arduino, plug and play, easy to use. Matching with Arduino controller, you can build a TDS detector easily to measure the TDS value of liquid.

This product supports 3.3 ~ 5.5V wide voltage input, and 0 ~ 2.3V analog voltage output, which makes it compatible with 5V or 3.3V control system or board. The excitation source is AC signal, which can effectively prevent the probe from polarization and prolong the life of the probe, meanwhile, increase the stability of the output signal. The TDS probe is waterproof, it can be immersed in water for long time measurement.

This product can be used in water quality application, such as domestic water, hydroponics. With this product, you can easily DIY a TDS detector to reflect the cleanliness of water to protect your health.

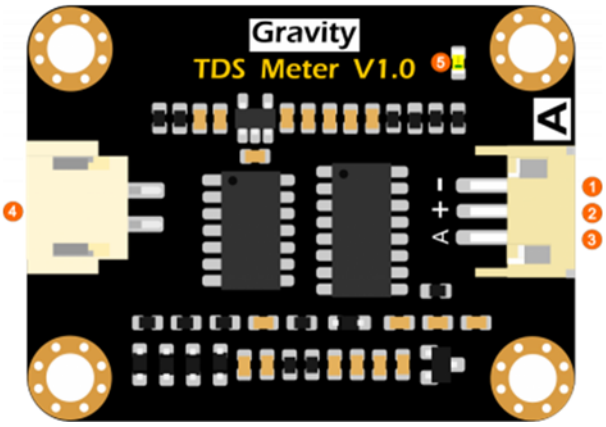
Attention:

1.The probe can not be used in water above 55 degrees centigrade.
2.The probe can not be left too close to the edge of the container, otherwise it will affect the reading.
3.The head and the cable of the probe are waterproof, but the connector and the signal transmitter board are not waterproof. Please be careful.

Specification

- **Signal Transmitter Board**
  Input Voltage: 3.3 ~ 5.5V
  Output Voltage: 0 ~ 2.3V
  Working Current: 3 ~ 6mA
  TDS Measurement Range: 0 ~ 1000ppm
  TDS Measurement Accuracy: ± 10% F.S. (25 ℃)
  Module Size: 42 * 32mm
  Module Interface: PH2.0-3P
  Electrode Interface: XH2.54-2P
- **TDS probe**
  Number of Needle: 2
  Total Length: 83cm
  Connection Interface: XH2.54-2P
  Colour: Black
  Other: Waterproof Probe

Board Overview



Analog TDS Sensor / Meter For Arduino

Num	Label	Description
1	-	Power GND(0V)
2	+	Power VCC(3.3 ~ 5.5V)
3	A	Analog Signal Output(0 ~ 2.3V)
4	TDS	TDS Probe Connector
5	LED	Power Indicator

Tutorial

This tutorial will show you how to measure the TDS value of the water. Please read this tutorial carefully, and pay attention to the steps and details.

The probe can not to be used in water above 55 degrees centigrade.
The probe can not be too close to the edge of the container, otherwise it will affect the reading.
The head and the cable of the probe are waterproof, but the connector and the signal transmitter board are not waterproof.Please pay attention to use.

Requirements

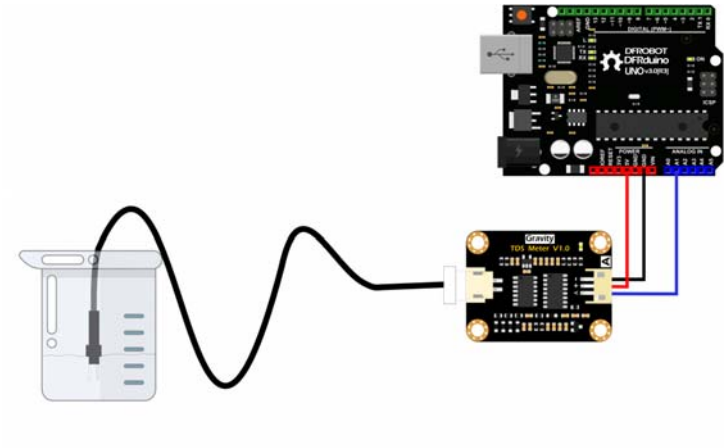
- **Hardware**
  DFRduino UNO R3 (or similar) x 1
  Analog TDS Sensor / Meter Module x 1
  TDS Probe x1
  Jumper Wires x3
  tested liquid x1

- **Software**

Arduino IDE (Version requirements: V1.0.x or V1.8.x), Click to Download Arduino IDE from Arduino®

https://www.arduino.cc/en/Main/Software%7C

Connection Diagram



Click here for production status of specific part numbers.

DS18B20

Programmable Resolution 1-Wire Digital Thermometer

General Description

The DS18B20 digital thermometer provides 9-bit to 12-bit Celsius temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18B20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. In addition, the DS18B20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply.

Each DS18B20 has a unique 64-bit serial code, which allows multiple DS18B20s to function on the same 1-Wire bus. Thus, it is simple to use one microprocessor to control many DS18B20s distributed over a large area. Applications that can benefit from this feature include HVAC environmental controls, temperature monitoring systems inside buildings, equipment, or machinery, and process monitoring and control systems.

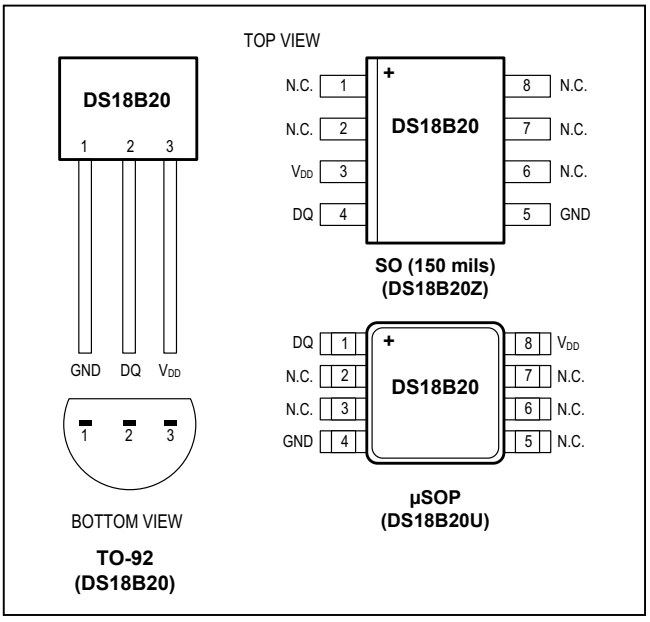
Applications

- Thermostatic Controls
- Industrial Systems
- Consumer Products
- Thermometers
- Thermally Sensitive Systems

Benefits and Features

- Unique 1-Wire® Interface Requires Only One Port Pin for Communication
- Reduce Component Count with Integrated Temperature Sensor and EEPROM
  - Measures Temperatures from -55°C to +125°C (-67°F to +257°F)
  - ±0.5°C Accuracy from -10°C to +85°C
  - Programmable Resolution from 9 Bits to 12 Bits
  - No External Components Required
- Parasitic Power Mode Requires Only 2 Pins for Operation (DQ and GND)
- Simplifies Distributed Temperature-Sensing Applications with Multidrop Capability
  - Each Device Has a Unique 64-Bit Serial Code Stored in On-Board ROM
- Flexible User-Definable Nonvolatile (NV) Alarm Settings with Alarm Search Command Identifies Devices with Temperatures Outside Programmed Limits
- Available in 8-Pin SO (150 mils), 8-Pin µSOP, and 3-Pin TO-92 Packages

Pin Configurations



***Ordering Information* appears at end of data sheet.**

1-Wire is a registered trademark of Maxim Integrated Products, Inc.

maxim integrated™

Absolute Maximum Ratings

Voltage Range on Any Pin Relative to Ground ....-0.5V to +6.0V
Operating Temperature Range..... -55°C to +125°C
Storage Temperature Range..... -55°C to +125°C
Solder Temperature.....Refer to the IPC/JEDEC J-STD-020 Specification.

These are stress ratings only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

DC Electrical Characteristics

(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Supply Voltage	V_{DD}	Local power (Note 1)		+3.0		+5.5	V
Pullup Supply Voltage	V_{PU}	Parasite power	(Notes 1, 2)	+3.0		+5.5	V
		Local power		+3.0		V_{DD}	
Thermometer Error	t_{ERR}	-10°C to +85°C	(Note 3)			±0.5	°C
		-30°C to +100°C				±1	
		-55°C to +125°C				±2	
Input Logic-Low	V_{IL}	(Notes 1, 4, 5)		-0.3		+0.8	V
Input Logic-High	V_{IH}	Local power	(Notes 1,6)	+2.2		The lower of 5.5 or V_{DD} + 0.3	V
		Parasite power		+3.0			
Sink Current	I_L	$V_{I/O}$ = 0.4V		4.0			mA
Standby Current	I_{DDS}	(Notes 7, 8)			750	1000	nA
Active Current	I_{DD}	V_{DD} = 5V (Note 9)			1	1.5	mA
DQ Input Current	I_{DQ}	(Note 10)			5		µA
Drift		(Note 11)			±0.2		°C

Note 1: All voltages are referenced to ground.
Note 2: The Pullup Supply Voltage specification assumes that the pullup device is ideal, and therefore the high level of the pullup is equal to V_{PU}. In order to meet the V_{IH} spec of the DS18B20, the actual supply rail for the strong pullup transistor must include margin for the voltage drop across the transistor when it is turned on; thus: $V_{PU\_ACTUAL} = V_{PU\_IDEAL} + V_{TRANSISTOR}$.
Note 3: See typical performance curve in Figure 1. Thermometer Error limits are 3-sigma values.
Note 4: Logic-low voltages are specified at a sink current of 4mA.
Note 5: To guarantee a presence pulse under low voltage parasite power conditions, V_{ILMAX} may have to be reduced to as low as 0.5V.
Note 6: Logic-high voltages are specified at a source current of 1mA.
Note 7: Standby current specified up to +70°C. Standby current typically is 3µA at +125°C.
Note 8: To minimize I_{DDS}, DQ should be within the following ranges: GND ≤ DQ ≤ GND + 0.3V or V_{DD} – 0.3V ≤ DQ ≤ V_{DD}.
Note 9: Active current refers to supply current during active temperature conversions or EEPROM writes.
Note 10: DQ line is high (“high-Z” state).
Note 11: Drift data is based on a 1000-hour stress test at +125°C with V_{DD} = 5.5V.

AC Electrical Characteristics–NV Memory

(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
NV Write Cycle Time	t_{WR}			2	10	ms
EEPROM Writes	N_{EEWR}	-55°C to +55°C	50k			writes
EEPROM Data Retention	t_{EEDR}	-55°C to +55°C	10			years

AC Electrical Characteristics

(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Temperature Conversion Time	t_{CONV}	9-bit resolution	(Note 12)			93.75	ms
		10-bit resolution				187.5	
		11-bit resolution				375	
		12-bit resolution				750	
Time to Strong Pullup On	t_{SPON}	Start convert T command issued				10	µs
Time Slot	t_{SLOT}	(Note 12)		60		120	µs
Recovery Time	t_{REC}	(Note 12)		1			µs
Write 0 Low Time	t_{LOW0}	(Note 12)		60		120	µs
Write 1 Low Time	t_{LOW1}	(Note 12)		1		15	µs
Read Data Valid	t_{RDV}	(Note 12)				15	µs
Reset Time High	t_{RSTH}	(Note 12)		480			µs
Reset Time Low	t_{RSTL}	(Notes 12, 13)		480			µs
Presence-Detect High	t_{PDHIGH}	(Note 12)		15		60	µs
Presence-Detect Low	t_{PDLOW}	(Note 12)		60		240	µs
Capacitance	$C_{IN/OUT}$					25	pF

Note 12: See the timing diagrams in Figure 2.
Note 13: Under parasite power, if t_{RSTL} > 960µs, a power-on reset can occur.

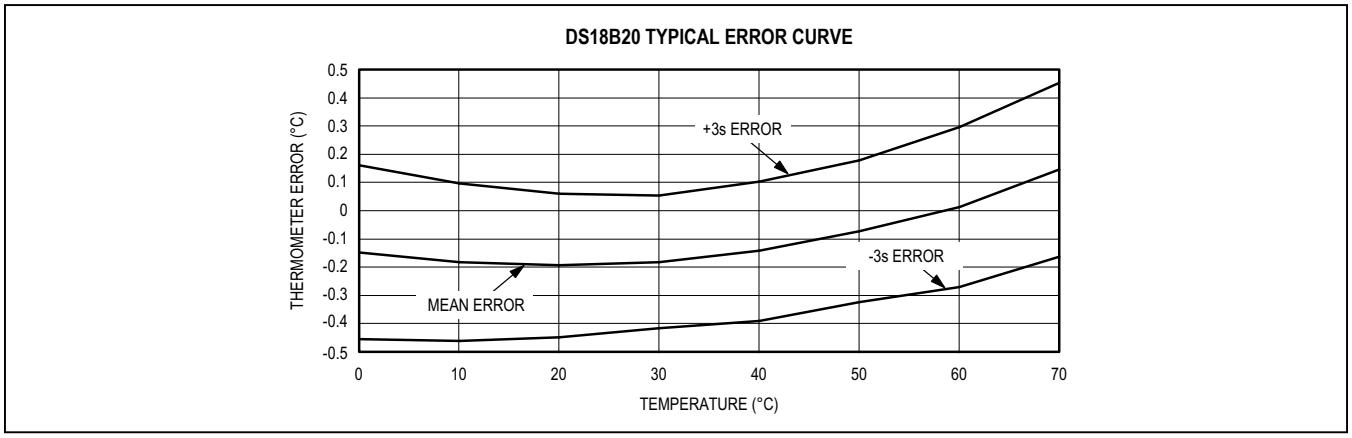


Figure 1. Typical Performance Curve

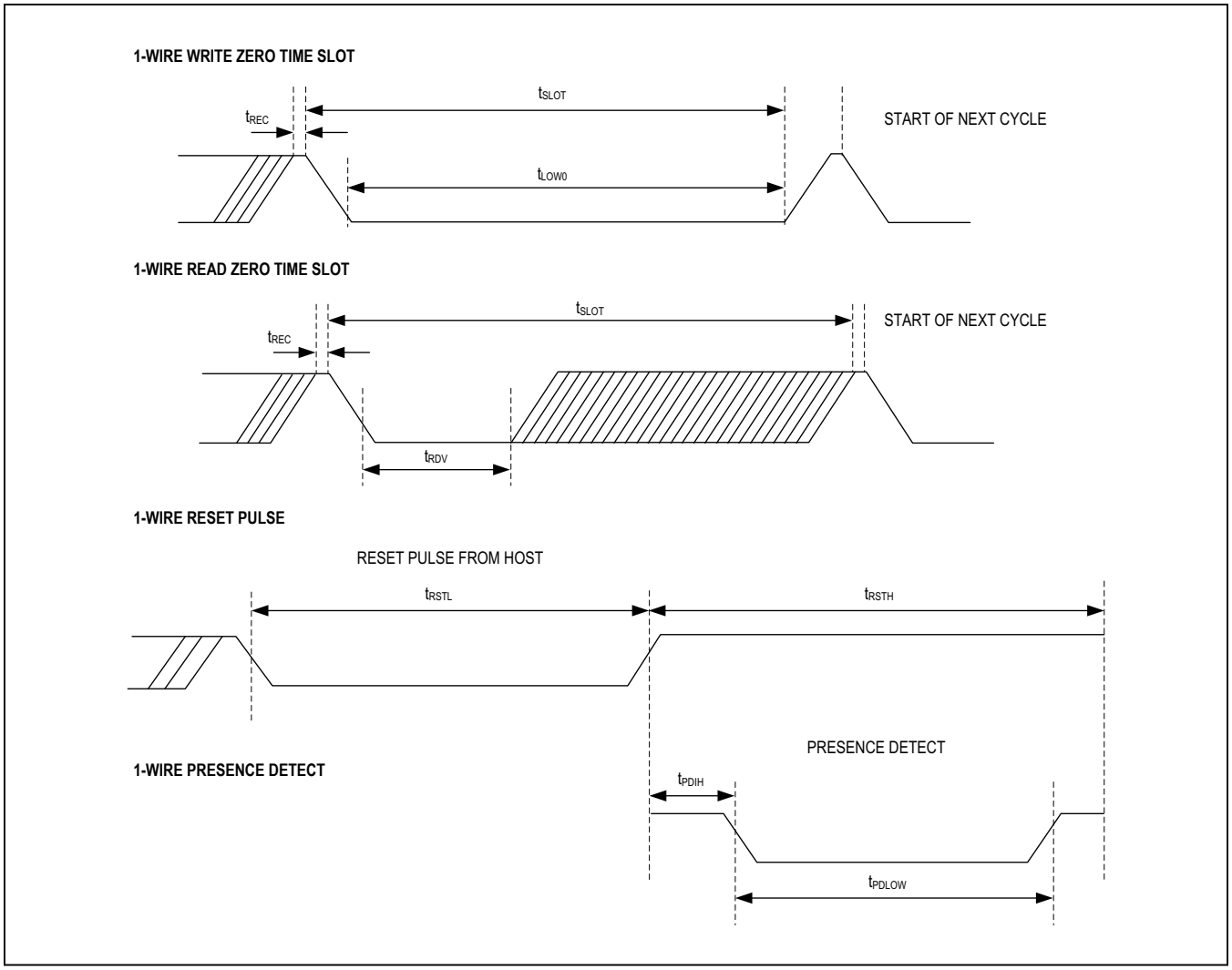


Figure 2. Timing Diagrams

Pin Description

PIN			NAME	FUNCTION
SO	µSOP	TO-92		
1, 2, 6, 7, 8	2, 3, 5, 6, 7	—	N.C.	No Connection
3	8	3	V_{DD}	Optional V_{DD}. V_{DD} must be grounded for operation in parasite power mode.
4	1	2	DQ	Data Input/Output. Open-drain 1-Wire interface pin. Also provides power to the device when used in parasite power mode (see the *Powering the DS18B20* section.)
5	4	1	GND	Ground

Overview

Figure 3 shows a block diagram of the DS18B20, and pin descriptions are given in the *Pin Description* table. The 64-bit ROM stores the device's unique serial code. The scratchpad memory contains the 2-byte temperature register that stores the digital output from the temperature sensor. In addition, the scratchpad provides access to the 1-byte upper and lower alarm trigger registers (T_H and T_L) and the 1-byte configuration register. The configuration register allows the user to set the resolution of the temperature-to-digital conversion to 9, 10, 11, or 12 bits. The T_H, T_L, and configuration registers are nonvolatile (EEPROM), so they will retain data when the device is powered down.

The DS18B20 uses Maxim's exclusive 1-Wire bus protocol that implements bus communication using one control signal. The control line requires a weak pullup resistor since all devices are linked to the bus via a 3-state or open-drain port (the DQ pin in the case of the DS18B20). In this bus system, the microprocessor (the master device) identifies and addresses devices on the bus using each device's unique 64-bit code. Because each device has a unique code, the number of devices that can be addressed on one bus is virtually unlimited. The 1-Wire bus protocol, including detailed explanations of the commands and "time slots," is covered in the *1-Wire Bus System* section.

Another feature of the DS18B20 is the ability to operate without an external power supply. Power is instead supplied through the 1-Wire pullup resistor through the DQ pin when the bus is high. The high bus signal also charges an internal capacitor (C_{PP}), which then supplies power to the device when the bus is low. This method of deriving power from the 1-Wire bus is referred to as "parasite power." As an alternative, the DS18B20 may also be powered by an external supply on V_{DD}.

Operation—Measuring Temperature

The core functionality of the DS18B20 is its direct-to-digital temperature sensor. The resolution of the temperature sensor is user-configurable to 9, 10, 11, or 12 bits, corresponding to increments of 0.5°C, 0.25°C, 0.125°C, and 0.0625°C, respectively. The default resolution at power-up is 12-bit. The DS18B20 powers up in a low-power idle state. To initiate a temperature measurement and A-to-D conversion, the master must issue a Convert T [44h] command. Following the conversion, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory and the DS18B20 returns to its idle state. If the DS18B20 is powered by an external supply, the master can issue "read time slots" (see the *1-Wire Bus System* section) after the Convert T command and the DS18B20 will respond by transmitting 0 while the temperature conversion is in progress and 1 when the conversion is done. If the DS18B20 is powered with parasite power, this notification technique cannot be used since the bus must be pulled high by a strong pullup during the entire temperature conversion. The bus requirements for parasite power are explained in detail in the *Powering the DS18B20* section.

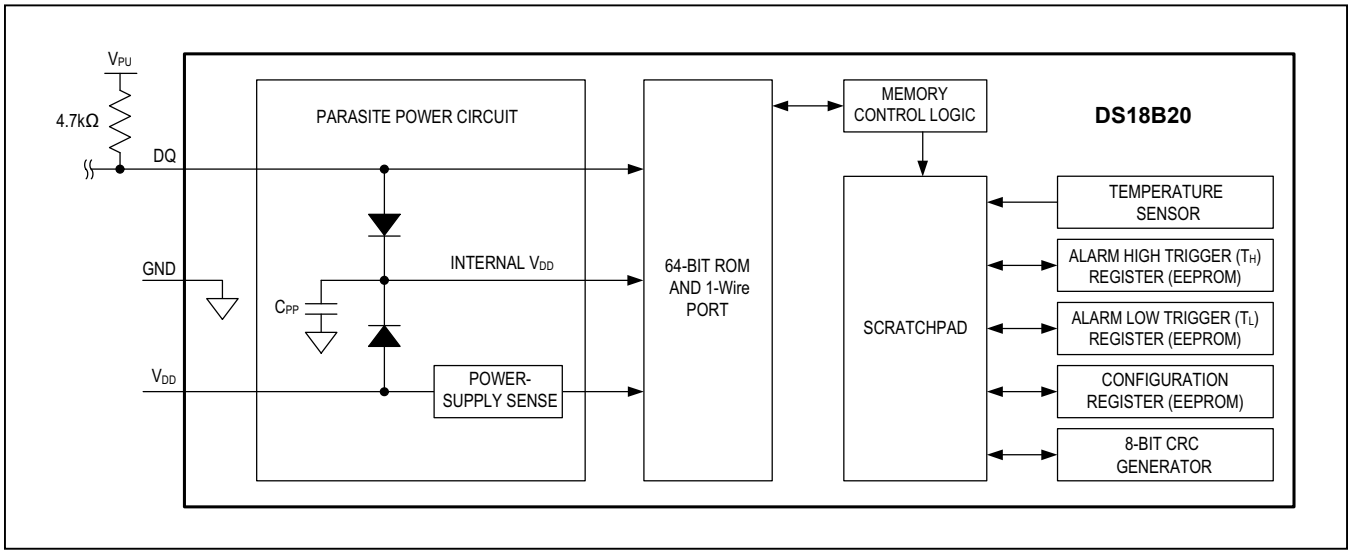


Figure 3. DS18B20 Block Diagram

The DS18B20 output temperature data is calibrated in degrees Celsius; for Fahrenheit applications, a lookup table or conversion routine must be used. The temperature data is stored as a 16-bit sign-extended two's complement number in the temperature register (see Figure 4). The sign bits (S) indicate if the temperature is positive or negative: for positive numbers S = 0 and for negative numbers S = 1. If the DS18B20 is configured for 12-bit resolution, all bits in the temperature register will contain valid data. For 11-bit resolution, bit 0 is undefined. For 10-bit resolution, bits 1 and 0 are undefined, and for 9-bit resolution bits 2, 1, and 0 are undefined. Table 1 gives examples of digital output data and the corresponding temperature reading for 12-bit resolution conversions.

Operation—Alarm Signaling

After the DS18B20 performs a temperature conversion, the temperature value is compared to the user-defined two's complement alarm trigger values stored in the 1-byte T_H and T_L registers (see Figure 5). The sign bit (S) indicates if the value is positive or negative: for positive numbers S = 0 and for negative numbers S = 1. The T_H and T_L registers are nonvolatile (EEPROM) so they will retain data when the device is powered down. T_H and T_L can be accessed through bytes 2 and 3 of the scratchpad as explained in the *Memory* section.

Only bits 11 through 4 of the temperature register are used in the T_H and T_L comparison since T_H and T_L are 8-bit registers. If the measured temperature is lower than

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	BIT 15	**BIT 14**	**BIT 13**	**BIT 12**	**BIT 11**	**BIT 10**	**BIT 9**	**BIT 8**
MS BYTE	S	S	S	S	S	2^6	2^5	2^4

S = SIGN

Figure 4. Temperature Register Format

Table 1. Temperature/Data Relationship

TEMPERATURE (°C)	DIGITAL OUTPUT (BINARY)	DIGITAL OUTPUT (HEX)
+125	0000 0111 1101 0000	07D0h
+85*	0000 0101 0101 0000	0550h
+25.0625	0000 0001 1001 0001	0191h
+10.125	0000 0000 1010 0010	00A2h
+0.5	0000 0000 0000 1000	0008h
0	0000 0000 0000 0000	0000h
-0.5	1111 1111 1111 1000	FFF8h
-10.125	1111 1111 0101 1110	FF5Eh
-25.0625	1111 1110 0110 1111	FE6Fh
-55	1111 1100 1001 0000	FC90h

**The power-on reset value of the temperature register is +85°C.*

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
S	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Figure 5. T_H and T_L Register Format

or equal to T_L or higher than or equal to T_H, an alarm condition exists and an alarm flag is set inside the DS18B20. This flag is updated after every temperature measurement; therefore, if the alarm condition goes away, the flag will be turned off after the next temperature conversion.

The master device can check the alarm flag status of all DS18B20s on the bus by issuing an Alarm Search [ECh] command. Any DS18B20s with a set alarm flag will respond to the command, so the master can determine exactly which DS18B20s have experienced an alarm condition. If an alarm condition exists and the T_H or T_L settings have changed, another temperature conversion should be done to validate the alarm condition.

Powering the DS18B20

The DS18B20 can be powered by an external supply on the V_{DD} pin, or it can operate in "parasite power" mode, which allows the DS18B20 to function without a local external supply. Parasite power is very useful for applications that require remote temperature sensing or that are very space constrained. Figure 3 shows the DS18B20's parasite-power control circuitry, which "steals" power from the 1-Wire bus via the DQ pin when the bus is high. The stolen charge powers the DS18B20 while the bus is high, and some of the charge is stored on the parasite power capacitor (C_{PP}) to provide power when the bus is low. When the DS18B20 is used in parasite power mode, the V_{DD} pin must be connected to ground.

In parasite power mode, the 1-Wire bus and CPP can provide sufficient current to the DS18B20 for most operations as long as the specified timing and voltage requirements are met (see the *DC Electrical Characteristics* and *AC Electrical Characteristics*). However, when the DS18B20 is performing temperature conversions or copying data from the scratchpad memory to EEPROM, the operating current can be as high as 1.5mA. This current can cause an unacceptable voltage drop across the weak 1-Wire pullup resistor and is more current than can be supplied by C_{PP}. To assure that the DS18B20 has sufficient supply current, it is necessary to provide a strong pullup on the 1-Wire bus whenever temperature conversions are taking place or data is being copied from the scratchpad to EEPROM. This can be accomplished by using a MOSFET to pull the bus directly to the rail as shown in Figure 6. The 1-Wire bus must be switched to the strong pullup within 10µs (max) after a Convert T [44h] or Copy Scratchpad [48h] command is issued, and the bus must be held high by the pullup for the duration of the conversion (t_{CONV}) or data transfer (t_{WR} = 10ms). No other activity can take place on the 1-Wire bus while the pullup is enabled.

The DS18B20 can also be powered by the conventional method of connecting an external power supply to the V_{DD} pin, as shown in Figure 7. The advantage of this method is that the MOSFET pullup is not required, and the 1-Wire bus is free to carry other traffic during the temperature conversion time.

The use of parasite power is not recommended for temperatures above +100°C since the DS18B20 may not be able to sustain communications due to the higher leakage currents that can exist at these temperatures. For applications in which such temperatures are likely, it is strongly recommended that the DS18B20 be powered by an external power supply.

In some situations the bus master may not know whether the DS18B20s on the bus are parasite powered or powered by external supplies. The master needs this information to determine if the strong bus pullup should be used during temperature conversions. To get this information, the master can issue a Skip ROM [CCh] command followed by a Read Power Supply [B4h] command followed by a "read time slot". During the read time slot, parasite powered DS18B20s will pull the bus low, and externally powered DS18B20s will let the bus remain high. If the bus is pulled low, the master knows that it must supply the strong pullup on the 1-Wire bus during temperature conversions.

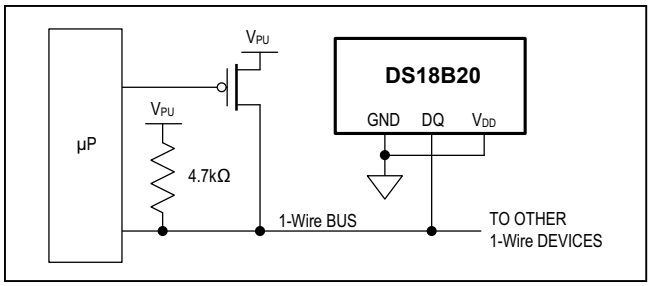


Figure 6. Supplying the Parasite-Powered DS18B20 During Temperature Conversions

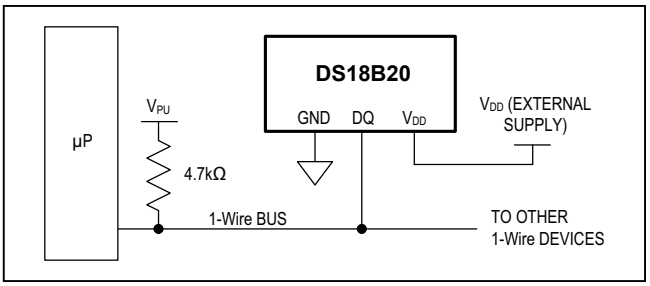


Figure 7. Powering the DS18B20 with an External Supply

64-BIT Lasered ROM code

Each DS18B20 contains a unique 64–bit code (see Figure 8) stored in ROM. The least significant 8 bits of the ROM code contain the DS18B20's 1-Wire family code: 28h. The next 48 bits contain a unique serial number. The most significant 8 bits contain a cyclic redundancy check (CRC) byte that is calculated from the first 56 bits of the ROM code. A detailed explanation of the CRC bits is provided in the *CRC Generation* section. The 64-bit ROM code and associated ROM function control logic allow the DS18B20 to operate as a 1-Wire device using the protocol detailed in the *1-Wire Bus System* section.

Memory

The DS18B20's memory is organized as shown in Figure 9. The memory consists of an SRAM scratchpad with nonvolatile EEPROM storage for the high and low alarm trigger registers (T_H and T_L) and configuration register. Note that if the DS18B20 alarm function is not used, the TH and TL registers can serve as general-purpose memory. All memory commands are described in detail in the *DS18B20 Function Commands* section.

Byte 0 and byte 1 of the scratchpad contain the LSB and the MSB of the temperature register, respectively. These bytes are read-only. Bytes 2 and 3 provide access to TH and TL registers. Byte 4 contains the configuration register data, which is explained in detail in the *Configuration Register* section. Bytes 5, 6, and 7 are reserved for internal use by the device and cannot be overwritten.

Byte 8 of the scratchpad is read-only and contains the CRC code for bytes 0 through 7 of the scratchpad. The DS18B20 generates this CRC using the method described in the *CRC Generation* section.

Data is written to bytes 2, 3, and 4 of the scratchpad using the Write Scratchpad [4Eh] command; the data must be transmitted to the DS18B20 starting with the least significant bit of byte 2. To verify data integrity, the scratchpad can be read (using the Read Scratchpad [BEh] command) after the data is written. When reading the scratchpad, data is transferred over the 1-Wire bus starting with the least significant bit of byte 0. To transfer the T_H, T_L and configuration data from the scratchpad to EEPROM, the master must issue the Copy Scratchpad [48h] command.

Data in the EEPROM registers is retained when the device is powered down; at power-up the EEPROM data is reloaded into the corresponding scratchpad locations. Data can also be reloaded from EEPROM to the scratchpad at any time using the Recall E^2 [B8h] command. The master can issue read time slots following the Recall E^2 command and the DS18B20 will indicate the status of the recall by transmitting 0 while the recall is in progress and 1 when the recall is done.

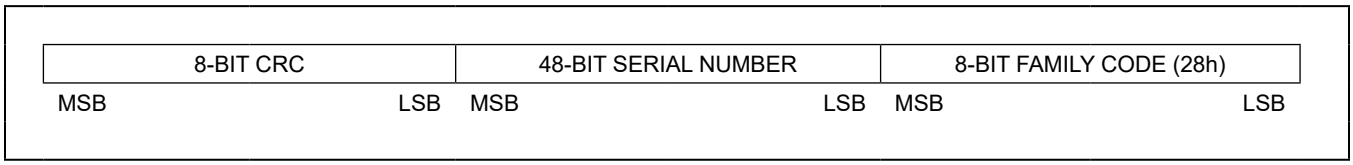


Figure 8. 64-Bit Lasered ROM Code

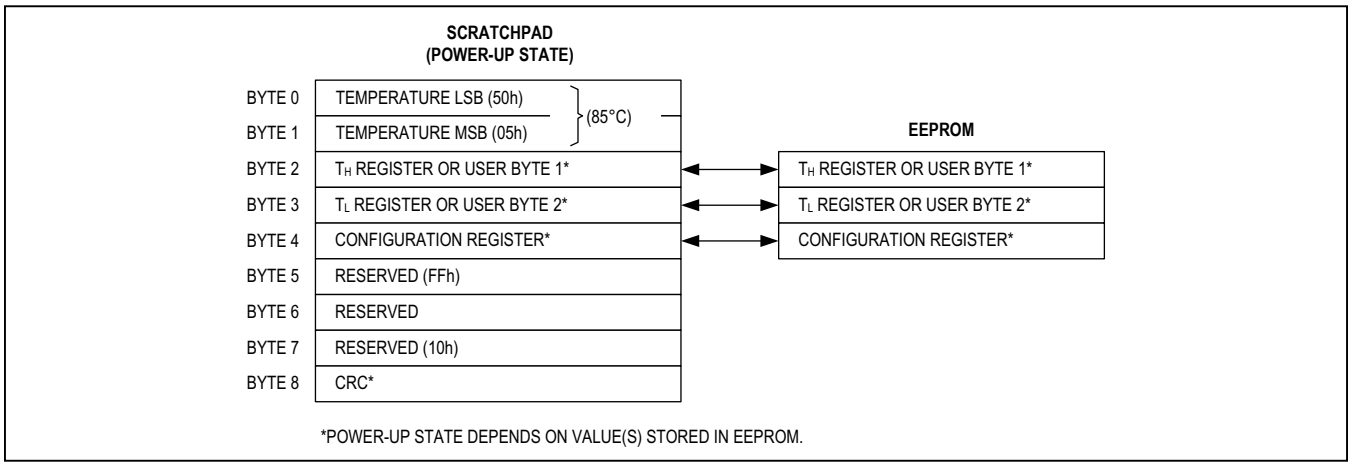


Figure 9. DS18B20 Memory Map

Configuration Register

Byte 4 of the scratchpad memory contains the configuration register, which is organized as illustrated in Figure 10. The user can set the conversion resolution of the DS18B20 using the R0 and R1 bits in this register as shown in Table 2. The power-up default of these bits is R0 = 1 and R1 = 1 (12-bit resolution). Note that there is a direct tradeoff between resolution and conversion time. Bit 7 and bits 0 to 4 in the configuration register are reserved for internal use by the device and cannot be overwritten.

CRC Generation

CRC bytes are provided as part of the DS18B20's 64-bit ROM code and in the 9th byte of the scratchpad memory. The ROM code CRC is calculated from the first 56 bits of the ROM code and is contained in the most significant byte of the ROM. The scratchpad CRC is calculated from the data stored in the scratchpad, and therefore it changes when the data in the scratchpad changes. The CRCs provide the bus master with a method of data validation when data is read from the DS18B20. To verify that data has been read correctly, the bus master must re-calculate the CRC from the received data and then compare this value to either the ROM code CRC (for ROM reads) or to the scratchpad CRC (for scratchpad reads). If the calculated CRC matches the read CRC, the data has been received error free. The comparison of CRC values and the decision to continue with an operation are determined entirely by the bus master. There is no circuitry inside the DS18B20 that prevents a command sequence from proceeding if the DS18B20 CRC (ROM or scratchpad) does not match the value generated by the bus master.

The equivalent polynomial function of the CRC (ROM or scratchpad) is:

$$CRC = X^8 + X^5 + X^4 + 1$$

The bus master can re-calculate the CRC and compare it to the CRC values from the DS18B20 using the polynomial generator shown in Figure 11. This circuit consists of a shift register and XOR gates, and the shift register bits are initialized to 0. Starting with the least significant bit of the ROM code or the least significant bit of byte 0 in the scratchpad, one bit at a time should shifted into the shift register. After shifting in the 56th bit from the ROM or the most significant bit of byte 7 from the scratchpad, the polynomial generator will contain the recalculated CRC. Next, the 8-bit ROM code or scratchpad CRC from the DS18B20 must be shifted into the circuit. At this point, if the re-calculated CRC was correct, the shift register will contain all 0s. Additional information about the Maxim 1-Wire cyclic redundancy check is available in *Application Note 27: Understanding and Using Cyclic Redundancy Checks with Maxim iButton Products*.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0	R1	R0	1	1	1	1	1

Figure 10. Configuration Register

Table 2. Thermometer Resolution Configuration

R1	R0	RESOLUTION (BITS)	MAX CONVERSION TIME	
0	0	9	93.75ms	(t_{CONV}/8)
0	1	10	187.5ms	(t_{CONV}/4)
1	0	11	375ms	(t_{CONV}/2)
1	1	12	750ms	(t_{CONV})

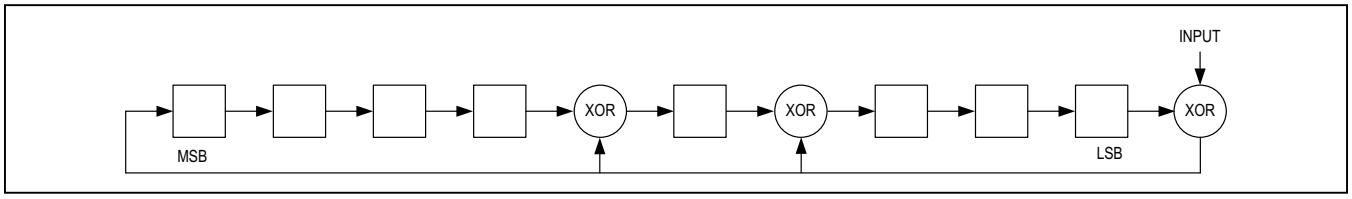


Figure 11. CRC Generator

1-Wire Bus System

The 1-Wire bus system uses a single bus master to control one or more slave devices. The DS18B20 is always a slave. When there is only one slave on the bus, the system is referred to as a “single-drop” system; the system is “multidrop” if there are multiple slaves on the bus.

All data and commands are transmitted least significant bit first over the 1-Wire bus.

The following discussion of the 1-Wire bus system is broken down into three topics: hardware configuration, transaction sequence, and 1-Wire signaling (signal types and timing).

Hardware Configuration

The 1-Wire bus has by definition only a single data line. Each device (master or slave) interfaces to the data line via an open-drain or 3-state port. This allows each device to “release” the data line when the device is not transmitting data so the bus is available for use by another device. The 1-Wire port of the DS18B20 (the DQ pin) is open drain with an internal circuit equivalent to that shown in Figure 12.

The 1-Wire bus requires an external pullup resistor of approximately 5kΩ; thus, the idle state for the 1-Wire bus is high. If for any reason a transaction needs to be suspended, the bus MUST be left in the idle state if the transaction is to resume. Infinite recovery time can occur between bits so long as the 1-Wire bus is in the inactive (high) state during the recovery period. If the bus is held low for more than 480µs, all components on the bus will be reset.

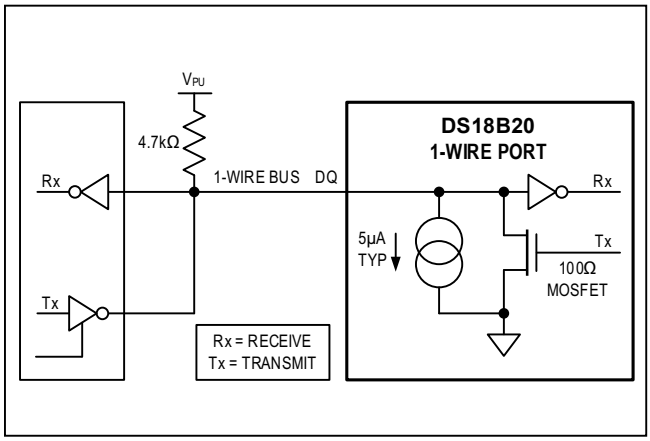


Figure 12. Hardware Configuration

Transaction Sequence

The transaction sequence for accessing the DS18B20 is as follows:

Step 1. Initialization

Step 2. ROM Command (followed by any required data exchange)

Step 3. DS18B20 Function Command (followed by any required data exchange)

It is very important to follow this sequence every time the DS18B20 is accessed, as the DS18B20 will not respond if any steps in the sequence are missing or out of order. Exceptions to this rule are the Search ROM [F0h] and Alarm Search [ECh] commands. After issuing either of these ROM commands, the master must return to Step 1 in the sequence.

Initialization

All transactions on the 1-Wire bus begin with an initialization sequence. The initialization sequence consists of a reset pulse transmitted by the bus master followed by presence pulse(s) transmitted by the slave(s). The presence pulse lets the bus master know that slave devices (such as the DS18B20) are on the bus and are ready to operate. Timing for the reset and presence pulses is detailed in the *1-Wire Signaling* section.

ROM Commands

After the bus master has detected a presence pulse, it can issue a ROM command. These commands operate on the unique 64-bit ROM codes of each slave device and allow the master to single out a specific device if many are present on the 1-Wire bus. These commands also allow the master to determine how many and what types of devices are present on the bus or if any device has experienced an alarm condition. There are five ROM commands, and each command is 8 bits long. The master device must issue an appropriate ROM command before issuing a DS18B20 function command. A flowchart for operation of the ROM commands is shown in Figure 13.

Search Rom [F0h]

When a system is initially powered up, the master must identify the ROM codes of all slave devices on the bus, which allows the master to determine the number of slaves and their device types. The master learns the ROM codes through a process of elimination that requires the master to perform a Search ROM cycle (i.e., Search ROM command followed by data exchange) as many times as necessary to identify all of the slave devices.

If there is only one slave on the bus, the simpler Read ROM [33h] command can be used in place of the Search ROM process. For a detailed explanation of the Search ROM procedure, refer to *Application Note 937: Book of iButton® Standards*. After every Search ROM cycle, the bus master must return to Step 1 (Initialization) in the transaction sequence.

Read Rom [33h]

This command can only be used when there is one slave on the bus. It allows the bus master to read the slave's 64-bit ROM code without using the Search ROM procedure. If this command is used when there is more than one slave present on the bus, a data collision will occur when all the slaves attempt to respond at the same time.

Match Rom [55H]

The match ROM command followed by a 64-bit ROM code sequence allows the bus master to address a specific slave device on a multidrop or single-drop bus. Only the slave that exactly matches the 64-bit ROM code sequence will respond to the function command issued by the master; all other slaves on the bus will wait for a reset pulse.

Skip Rom [CCh]

The master can use this command to address all devices on the bus simultaneously without sending out any ROM code information. For example, the master can make all DS18B20s on the bus perform simultaneous temperature conversions by issuing a Skip ROM command followed by a Convert T [44h] command.

Note that the Read Scratchpad [BEh] command can follow the Skip ROM command only if there is a single slave device on the bus. In this case, time is saved by allowing the master to read from the slave without sending the device's 64-bit ROM code. A Skip ROM command followed by a Read Scratchpad command will cause a data collision on the bus if there is more than one slave since multiple devices will attempt to transmit data simultaneously.

Alarm Search [ECh]

The operation of this command is identical to the operation of the Search ROM command except that only slaves with a set alarm flag will respond. This command allows the master device to determine if any DS18B20s experienced an alarm condition during the most recent temperature conversion. After every Alarm Search cycle (i.e., Alarm Search command followed by data exchange), the bus master must return to Step 1 (Initialization) in the transaction sequence. See the *Operation—Alarm Signaling* section for an explanation of alarm flag operation.

DS18B20 Function Commands

After the bus master has used a ROM command to address the DS18B20 with which it wishes to communicate, the master can issue one of the DS18B20 function commands. These commands allow the master to write to and read from the DS18B20's scratchpad memory, initiate temperature conversions and determine the power supply mode. The DS18B20 function commands, which are described below, are summarized in Table 3 and illustrated by the flowchart in Figure 14.

Convert T [44h]

This command initiates a single temperature conversion. Following the conversion, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory and the DS18B20 returns to its low-power idle state. If the device is being used in parasite power mode, within 10µs (max) after this command is issued the master must enable a strong pullup on the 1-Wire bus for the duration of the conversion (t_{CONV}) as described in the *Powering the DS18B20* section. If the DS18B20 is powered by an external supply, the master can issue read time slots after the Convert T command and the DS18B20 will respond by transmitting a 0 while the temperature conversion is in progress and a 1 when the conversion is done. In parasite power mode this notification technique cannot be used since the bus is pulled high by the strong pullup during the conversion.

Write Scratchpad [4Eh]

This command allows the master to write 3 bytes of data to the DS18B20's scratchpad. The first data byte is written into the T_H register (byte 2 of the scratchpad), the second byte is written into the T_L register (byte 3), and the third byte is written into the configuration register (byte 4). Data must be transmitted least significant bit first. All three bytes MUST be written before the master issues a reset, or the data may be corrupted.

Read Scratchpad [BEh]

This command allows the master to read the contents of the scratchpad. The data transfer starts with the least significant bit of byte 0 and continues through the scratchpad until the 9th byte (byte 8 – CRC) is read. The master may issue a reset to terminate reading at any time if only part of the scratchpad data is needed.

iButton is a registered trademark of Maxim Integrated Products, Inc.

Copy Scratchpad [48h]

This command copies the contents of the scratchpad T_H, T_L and configuration registers (bytes 2, 3 and 4) to EEPROM. If the device is being used in parasite power mode, within 10µs (max) after this command is issued the master must enable a strong pullup on the 1-Wire bus for at least 10ms as described in the *Powering the DS18B20* section.

Recall E^2 [B8h]

This command recalls the alarm trigger values (T_H and T_L) and configuration data from EEPROM and places the data in bytes 2, 3, and 4, respectively, in the scratchpad memory. The master device can issue read time slots following the Recall E^2 command and the DS18B20 will indicate the status of the recall by transmitting 0 while the recall is in progress and 1 when the recall is done. The recall operation happens automatically at power-up, so valid data is available in the scratchpad as soon as power is applied to the device.

Read Power Supply [B4h]

The master device issues this command followed by a read time slot to determine if any DS18B20s on the bus are using parasite power. During the read time slot, parasite powered DS18B20s will pull the bus low, and externally powered DS18B20s will let the bus remain high. See the *Powering the DS18B20* section for usage information for this command.

Table 3. DS18B20 Function Command Set

COMMAND	DESCRIPTION	PROTOCOL	1-Wire BUS ACTIVITY AFTER COMMAND IS ISSUED	NOTES
TEMPERATURE CONVERSION COMMANDS				
Convert T	Initiates temperature conversion.	44h	DS18B20 transmits conversion status to master (not applicable for parasite-powered DS18B20s).	1
MEMORY COMMANDS				
Read Scratchpad	Reads the entire scratchpad including the CRC byte.	BEh	DS18B20 transmits up to 9 data bytes to master.	2
Write Scratchpad	Writes data into scratchpad bytes 2, 3, and 4 (T_H, T_L, and configuration registers).	4Eh	Master transmits 3 data bytes to DS18B20.	3
Copy Scratchpad	Copies T_H, T_L, and configuration register data from the scratchpad to EEPROM.	48h	None	1
Recall E^2	Recalls T_H, T_L, and configuration register data from EEPROM to the scratchpad.	B8h	DS18B20 transmits recall status to master.	
Read Power Supply	Signals DS18B20 power supply mode to the master.	B4h	DS18B20 transmits supply status to master.	

Note 1: For parasite-powered DS18B20s, the master must enable a strong pullup on the 1-Wire bus during temperature conversions and copies from the scratchpad to EEPROM. No other bus activity may take place during this time.
Note 2: The master can interrupt the transmission of data at any time by issuing a reset.
Note 3: All three bytes must be written before a reset is issued.

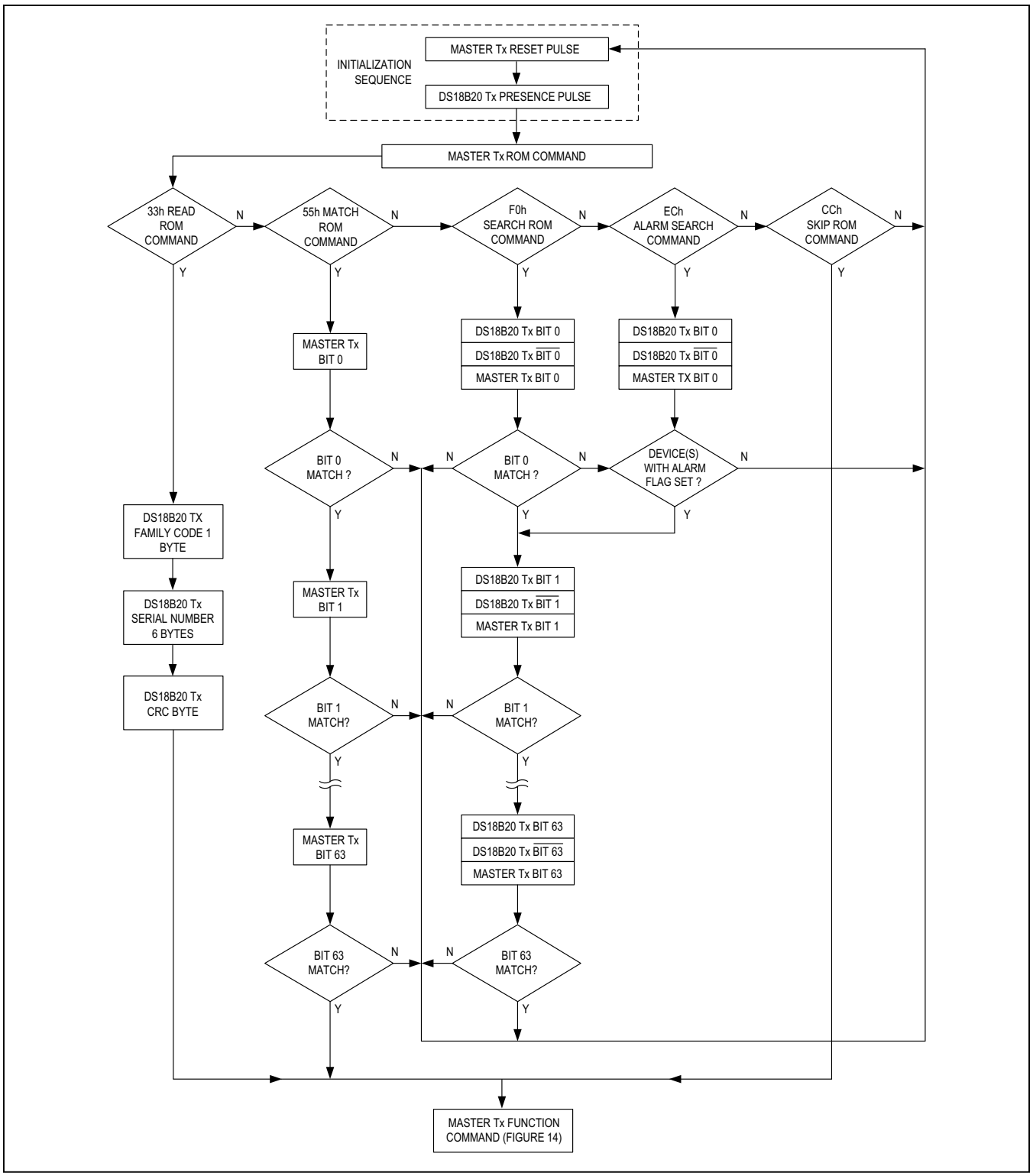


Figure 13. ROM Commands Flowchart

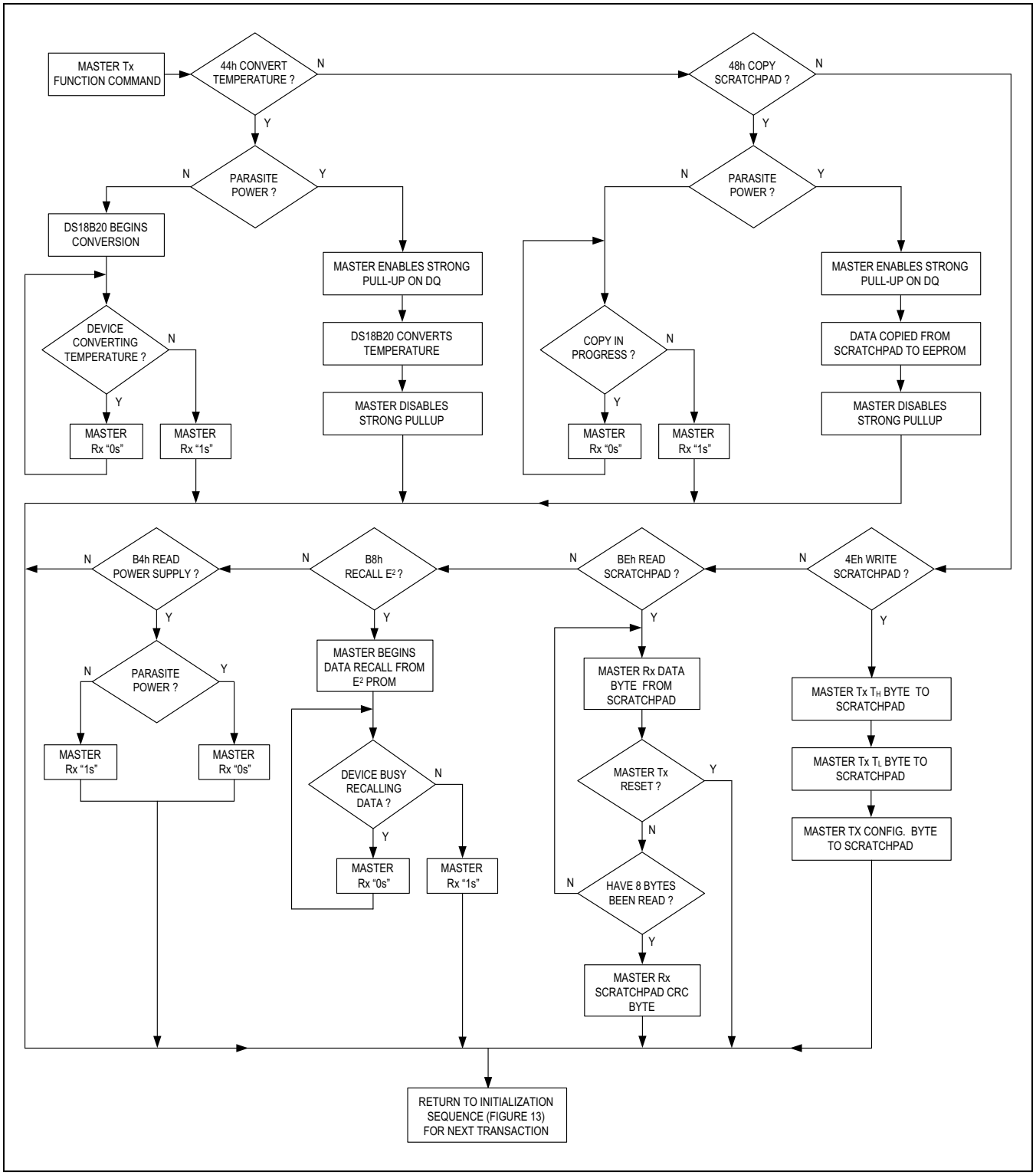


Figure 14. DS18B20 Function Commands Flowchart

1-Wire Signaling

The DS18B20 uses a strict 1-Wire communication protocol to ensure data integrity. Several signal types are defined by this protocol: reset pulse, presence pulse, write 0, write 1, read 0, and read 1. The bus master initiates all these signals, with the exception of the presence pulse.

Initialization Procedure—Reset And Presence Pulses

All communication with the DS18B20 begins with an initialization sequence that consists of a reset pulse from the master followed by a presence pulse from the DS18B20. This is illustrated in Figure 15. When the DS18B20 sends the presence pulse in response to the reset, it is indicating to the master that it is on the bus and ready to operate.

During the initialization sequence the bus master transmits (T_X) the reset pulse by pulling the 1-Wire bus low for a minimum of 480µs. The bus master then releases the bus and goes into receive mode (R_X). When the bus is released, the 5kΩ pullup resistor pulls the 1-Wire bus high. When the DS18B20 detects this rising edge, it waits 15µs to 60µs and then transmits a presence pulse by pulling the 1-Wire bus low for 60µs to 240µs.

Read/Write Time Slots

The bus master writes data to the DS18B20 during write time slots and reads data from the DS18B20 during read time slots. One bit of data is transmitted over the 1-Wire bus per time slot.

Write Time Slots

There are two types of write time slots: “Write 1” time slots and “Write 0” time slots. The bus master uses a Write 1 time slot to write a logic 1 to the DS18B20 and a Write 0 time slot to write a logic 0 to the DS18B20. All write time slots must be a minimum of 60µs in duration with a minimum of a 1µs recovery time between individual write slots. Both types of write time slots are initiated by the master pulling the 1-Wire bus low (see Figure 14).

To generate a Write 1 time slot, after pulling the 1-Wire bus low, the bus master must release the 1-Wire bus within 15µs. When the bus is released, the 5kΩ pullup resistor will pull the bus high. To generate a Write 0 time slot, after pulling the 1-Wire bus low, the bus master must continue to hold the bus low for the duration of the time slot (at least 60µs).

The DS18B20 samples the 1-Wire bus during a window that lasts from 15µs to 60µs after the master initiates the write time slot. If the bus is high during the sampling window, a 1 is written to the DS18B20. If the line is low, a 0 is written to the DS18B20.

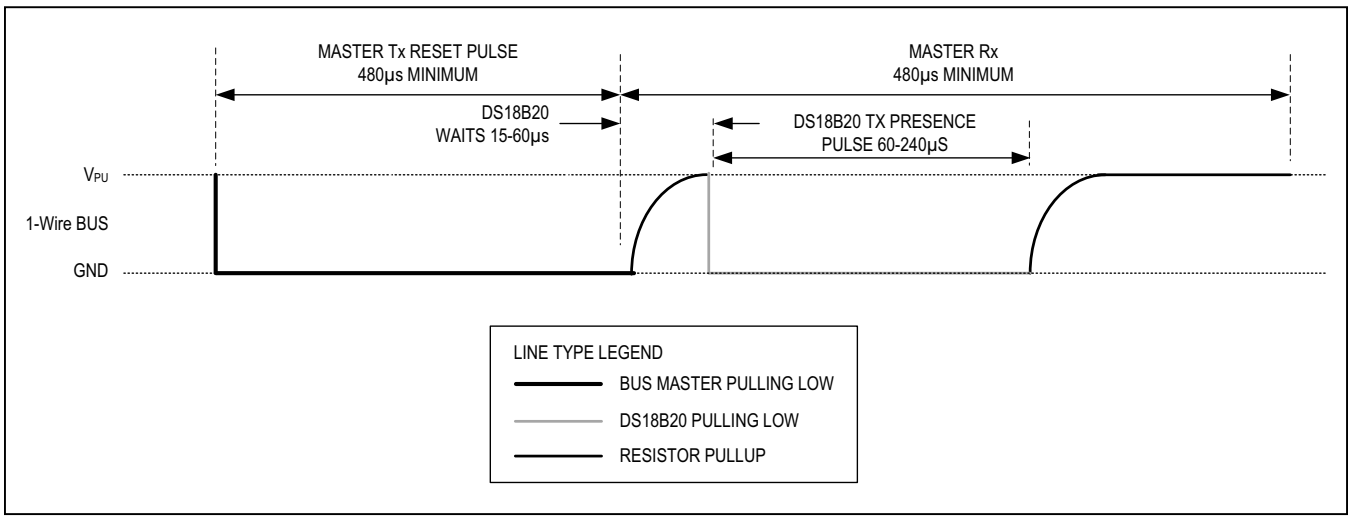


Figure 15. Initialization Timing

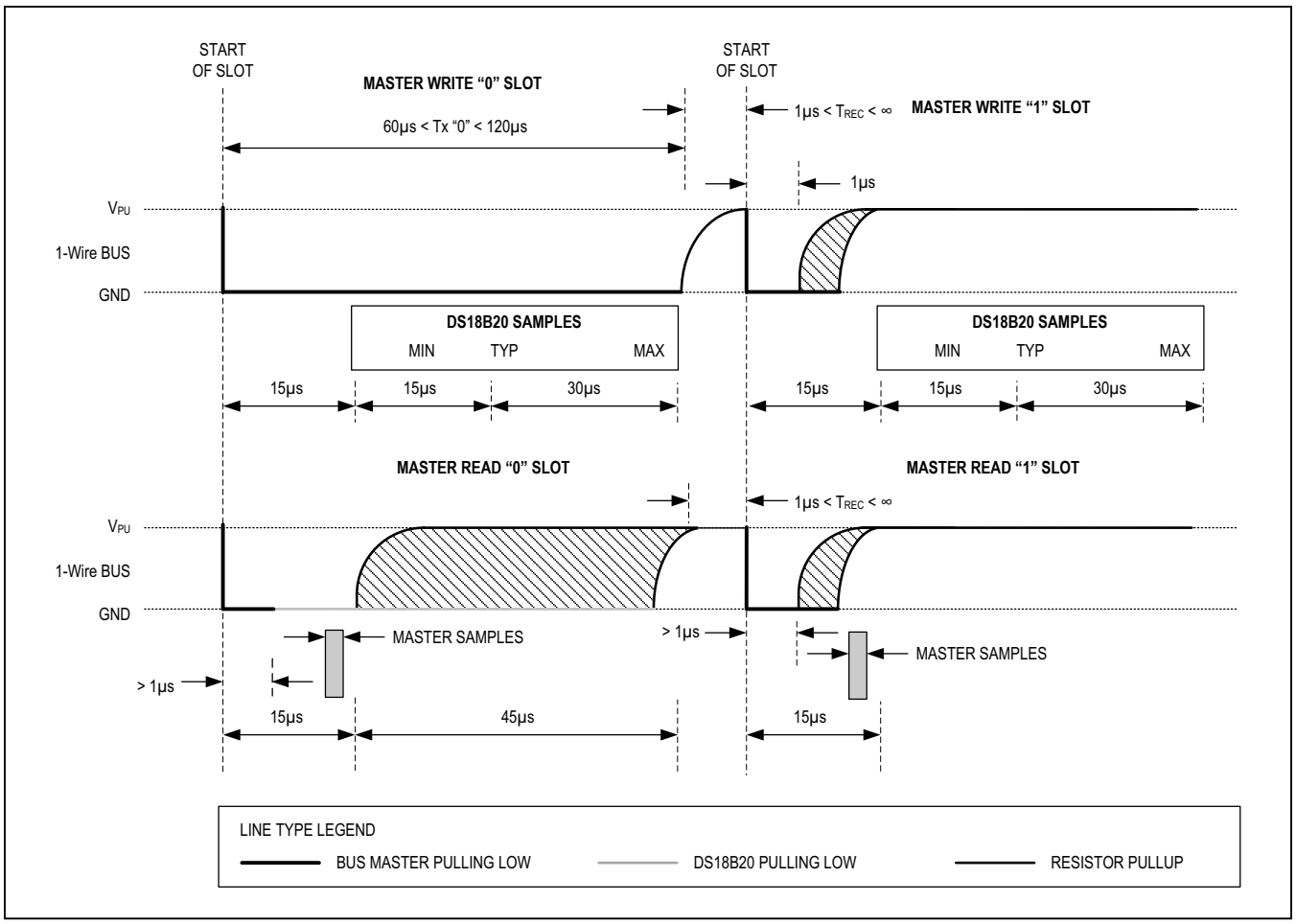


Figure 16. Read/Write Time Slot Timing Diagram

Read Time Slots

The DS18B20 can only transmit data to the master when the master issues read time slots. Therefore, the master must generate read time slots immediately after issuing a Read Scratchpad [BEh] or Read Power Supply [B4h] command, so that the DS18B20 can provide the requested data. In addition, the master can generate read time slots after issuing Convert T [44h] or Recall E^2 [B8h] commands to find out the status of the operation as explained in the *DS18B20 Function Commands* section.

All read time slots must be a minimum of 60µs in duration with a minimum of a 1µs recovery time between slots. A read time slot is initiated by the master device pulling the 1-Wire bus low for a minimum of 1µs and then releasing the bus (see Figure 16). After the master initiates the read time slot, the DS18B20 will begin transmitting a 1 or 0 on bus. The DS18B20 transmits a 1 by leaving the bus high and transmits a 0 by pulling the bus low. When transmitting a 0, the DS18B20 will release the bus by the end of the time slot, and the bus will be pulled back to its high idle state by the pullup resister. Output data from the DS18B20 is valid for 15µs after the falling edge that initiated the read time slot. Therefore, the master must release the bus and then sample the bus state within 15µs from the start of the slot.

Figure 17 illustrates that the sum of T_{INIT}, T_{RC}, and T_{SAMPLE} must be less than 15µs for a read time slot. Figure 18 shows that system timing margin is maximized by keeping T_{INIT} and T_{RC} as short as possible and by locating the master sample time during read time slots towards the end of the 15µs period.

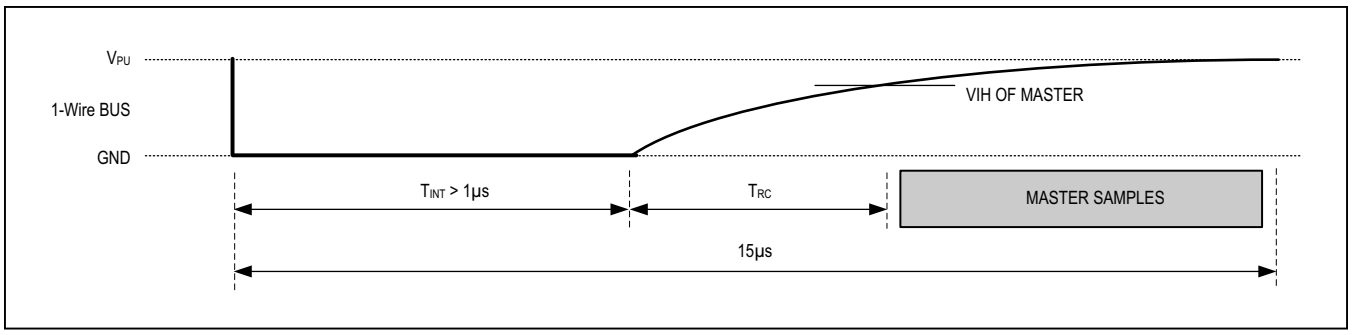


Figure 17. Detailed Master Read 1 Timing

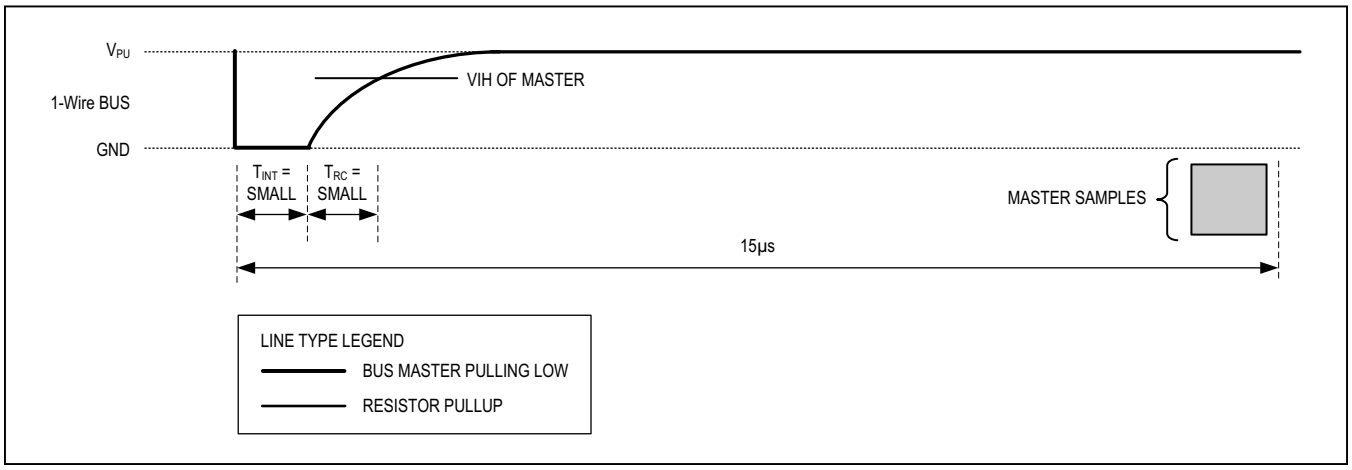


Figure 18. Recommended Master Read 1 Timing

Related Application Notes

The following application notes can be applied to the DS18B20 and are available at www.maximintegrated.com.

Application Note 27: Understanding and Using Cyclic Redundancy Checks with Maxim iButton Products

Application Note 122: Using Dallas' 1-Wire ICs in 1-Cell Li-Ion Battery Packs with Low-Side N-Channel Safety FETs Master

Application Note 126: 1-Wire Communication Through Software

Application Note 162: Interfacing the DS18x20/DS1822 1-Wire Temperature Sensor in a Microcontroller Environment

Application Note 208: Curve Fitting the Error of a Bandgap-Based Digital Temperature Sensor

Application Note 2420: 1-Wire Communication with a Microchip PICmicro Microcontroller

Application Note 3754: Single-Wire Serial Bus Carries Isolated Power and Data

Sample 1-Wire subroutines that can be used in conjunction with A*pplication Note 74: Reading and Writing iButtons via Serial Interfaces* can be downloaded from the Maxim website.

DS18B20 Operation Example 1

In this example there are multiple DS18B20s on the bus and they are using parasite power. The bus master initiates a temperature conversion in a specific DS18B20 and then reads its scratchpad and recalculates the CRC to verify the data.

MASTER MODE	DATA (LSB FIRST)	COMMENTS
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20s respond with presence pulse.
Tx	55h	Master issues Match ROM command.
Tx	64-bit ROM code	Master sends DS18B20 ROM code.
Tx	44h	Master issues Convert T command.
Tx	DQ line held high by strong pullup	Master applies strong pullup to DQ for the duration of the conversion (t_{CONV}).
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20s respond with presence pulse.
Tx	55h	Master issues Match ROM command.
Tx	64-bit ROM code	Master sends DS18B20 ROM code.
Tx	BEh	Master issues Read Scratchpad command.
Rx	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.

DS18B20 Operation Example 2

In this example there is only one DS18B20 on the bus and it is using parasite power. The master writes to the TH, TL, and configuration registers in the DS18B20 scratchpad and then reads the scratchpad and recalculates the CRC to verify the data. The master then copies the scratchpad contents to EEPROM.

MASTER MODE	DATA (LSB FIRST)	COMMENTS
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	4Eh	Master issues Write Scratchpad command.
Tx	3 data bytes	Master sends three data bytes to scratchpad (T_H, T_L, and config).
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	BEh	Master issues Read Scratchpad command.
Rx	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	48h	Master issues Copy Scratchpad command.
Tx	DQ line held high by strong pullup	Master applies strong pullup to DQ for at least 10ms while copy operation is in progress.

Ordering Information

PART	TEMP RANGE	PIN-PACKAGE	TOP MARK
DS18B20	-55°C to +125°C	3 TO-92	18B20
DS18B20+	-55°C to +125°C	3 TO-92	18B20
DS18B20/T&R	-55°C to +125°C	3 TO-92 (2000 Piece)	18B20
DS18B20+T&R	-55°C to +125°C	3 TO-92 (2000 Piece)	18B20
DS18B20-SL/T&R	-55°C to +125°C	3 TO-92 (2000 Piece)*	18B20
DS18B20-SL+T&R	-55°C to +125°C	3 TO-92 (2000 Piece)*	18B20
DS18B20U	-55°C to +125°C	8 FSOP	18B20
DS18B20U+	-55°C to +125°C	8 FSOP	18B20
DS18B20U/T&R	-55°C to +125°C	8 FSOP (3000 Piece)	18B20
DS18B20U+T&R	-55°C to +125°C	8 FSOP (3000 Piece)	18B20
DS18B20Z	-55°C to +125°C	8 SO	DS18B20
DS18B20Z+	-55°C to +125°C	8 SO	DS18B20
DS18B20Z/T&R	-55°C to +125°C	8 SO (2500 Piece)	DS18B20
DS18B20Z+T&R	-55°C to +125°C	8 SO (2500 Piece)	DS18B20

+Denotes a lead-free package. A "+" will appear on the top mark of lead-free packages.

T&R = Tape and reel.

**TO-92 packages in tape and reel can be ordered with straight or formed leads. Choose "SL" for straight leads. Bulk TO-92 orders are straight leads only.*

Revision History

REVISION DATE	DESCRIPTION	PAGES CHANGED
3/1/07	In the Absolute Maximum Ratings section, removed the reflow oven temperature value of +220°C. Reference to JEDEC specification for reflow remains.	19
10/12/07	In the *Operation—Alarm Signaling* section, added “or equal to” in the description for a TH alarm condition	5
	In the *Memory* section, removed incorrect text describing memory.	7
	In the *Configuration Register* section, removed incorrect text describing configuration register.	8
4/22/08	In the *Ordering Information* table, added TO-92 straight-lead packages and included a note that the TO-92 package in tape and reel can be ordered with either formed or straight leads.	2
1/15	Updated *Benefits and Features* section	1
09/18	Updated *DC Electrical Characteristics* table	2
7/19	Updated Figure 12	10

PH meter(SKU: SEN0161)

Analog pH Meter Kit SKU: SEN0161

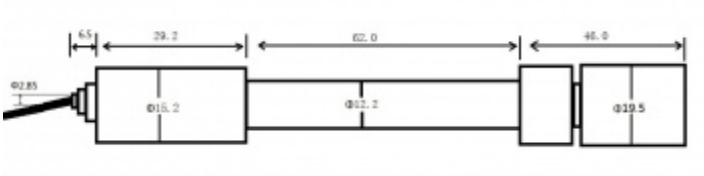
Analog pH Meter Kit SKU: SEN0169

Contents

Introduction

Need to measure water quality and other parameters but haven't got any low cost pH meter? Find it difficult to use with Arduino? Here comes an analog pH meter, specially designed for Arduino controllers and has built-in simple, convenient and practical connection and features. It has an LED which works as the Power Indicator, a BNC connector and PH2.0 sensor interface. You can just connect the pH sensor with BNC connector, and plug the PH2.0 interface into any analog input on Arduino controller to read pH value easily.

Specification



SEN0161 dimension

- Module Power: 5.00V
- Circuit Board Size: 43mm×32mm
- pH Measuring Range: 0-14
- Measuring Temperature: 0-60 ℃
- Accuracy: ± 0.1pH (25 ℃)
- Response Time: ≤ 1min
- pH Sensor with BNC Connector
- PH2.0 Interface (3 foot patch)
- Gain Adjustment Potentiometer
- Power Indicator LED

Precautions

- Before and after use of the pH electrode every time, you need to use (pure)water to clean it.
- The electrode plug should be kept clean and dry in case of short circuit.
- **Preservation**: Electrode reference preservation solution is the **3N KCL** solution.
- Measurement should be avoided staggered pollution between solutions, so as not to affect the accuracy of measurement.
- Electrode blub or sand core is defiled which will make PTS decline, slow response. So, it should be based on the characteristics of the pollutant, adapted to the cleaning solution, the electrode performance recovery.

- Electrode when in use, the ceramic sand core and liquid outlet rubber ring should be removed, in order to make salt bridge solution to maintain a certain velocity.

NOTE: **Differences between the probes, SEN0161 and SEN0169**

Their usages/ specifications are almost the same. The differences locates at

Long-firing Operation: SEN0169 supports, while SEN0161 NOT, i.e. you can not immerse SEN0161 in water for Continuous Testing.

Life Span: In 25 ℃, pure water, do Continuous Testing with them both, SEN0169 can work two years, while SEN0161 can only last for 6 months. And just for reference, if put them in turbid, strongly acid and alkali solution, 25℃, the life span would drop to one year (SEN0169), 1 month(or shorter, SEN0161).
Tempreture, pH, turbidity of the water effect the probe life span a lot.

Waterproof: You can immerse the whole probe SEN0169 into the water, while you can only immerse the front part of the probe SEN0161, the electrode glass bulb, into water, the rear part, from the white shell to the cable, MUST NOT be under water.

Strongly Acid and Alkali: SEN0169 are preferred for strongly acid and alkali test. And if your testing range is usually within pH6~8, then SEN0161 is capable for that.

pH Electrode Characteristics

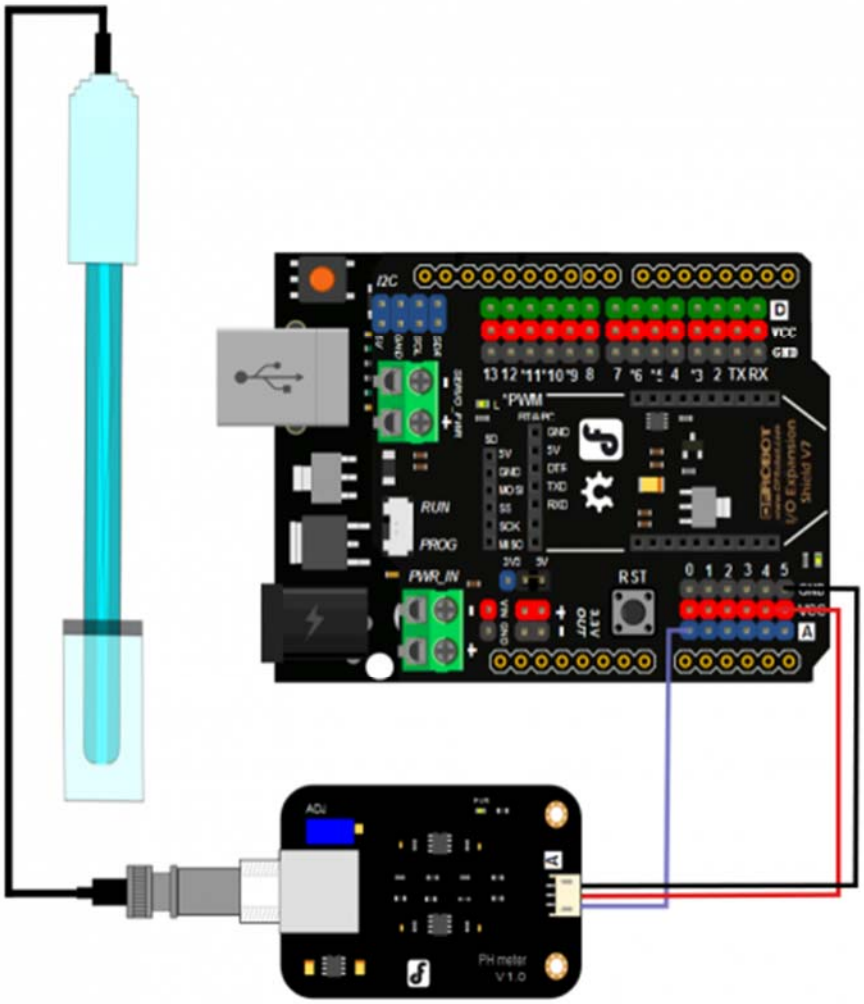
The output of pH electrode is Millivolts, and the pH value of the relationship is shown as follows (25 ℃):

VOLTAGE （mV）	pH value	VOLTAGE （mV）	pH value
414.12	0.00	-414.12	14.00
354.96	1.00	-354.96	13.00
295.80	2.00	-295.80	12.00
236.64	3.00	-236.64	11.00
177.48	4.00	-177.48	10.00
118.32	5.00	-118.32	9.00
59.16	6.00	-59.16	8.00
0.00	7.00	0.00	7.00

NOTE: It is normal that if your reading is much different with the table since you are not reading from the electrode directly but from the voltage adapter, it has converted the original voltage (-5V ~ +5V) to Arduino compatible voltage, i.e. 0 ~ 5V. See the discussion on Forum.

Usage

Connecting Diagram

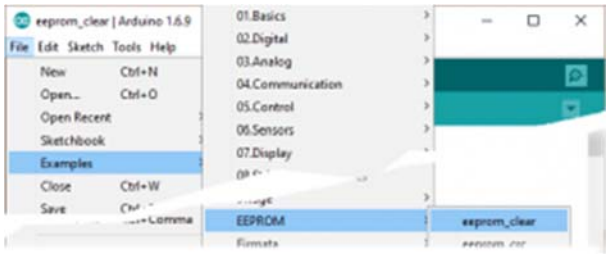


NOTE:

Before you insert the pH probe into one solution from another, or after you finish using the sensor, you must wash the pH electrode with pure water everytime (distilled water is the best)!
The closer **power supply** to +5.00V, the more accurate pH readings you could get.
You have to immerse the pH probe into stationary solution instead of the running one to get relative stable pH readings.
How long should it be under the solution? It depends on the pH value, the closer to neutral solution (pH = 7.00), the longer it will take. As we tested in water pH = 6.0, the blue one costs 6 minutes, and in standard Acid/ Alkali (4.00/ 10.00) solutions, it only needs 10 seconds.

Method 1. Software Calibration

The software calibration is easier than the next part - Hardware Calibration through the Potentiometer. Because it writes the calibration values into Arduino's EEPROM, so you can calibrate once for all if you won't replace your Arduino. It uses mathematical method that to draw a line using two points, i.e. using the Acid standard solution, pH = 4.00 and alkaline pH = 10.00 or 9.18 to draw the linear relation between the voltage and the pH value.



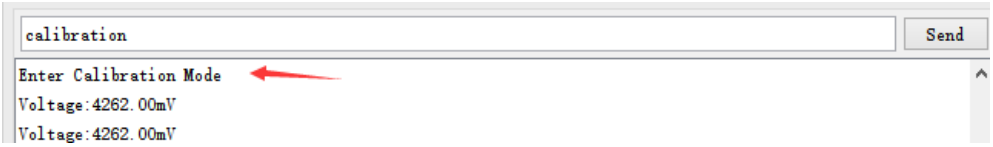
For NOTE 3. Arduino sample sketch "**EEPROM Clear**"

NOTE:

During the calibration (from step 4 to step 7), **power outage** should be avoided, or you will have to start over from step 4.
Software Calibration has nothing to do with the **potentiometer** on the adapter. Especially after you finished the calibration, you should never adjust the potentiometer, or you should start over. Moreover, considering the mechanical vibration might interfere the potentiometer value, you could seal it by Hot Melt Adhesive.
If you want to try Hardware Calibration, you'd better reset the EEPROM setting by uploading the Arduino IDE sample sketch "**EEPROM Clear**" as shown as the right hand picture.

Steps

1. Wiring the pH probe, pH meter adapter (the little PCB board) and Arduino UNO as the Diagram section above.
2. Upload the sample code "Software Calibration" below to UNO.
3. Open Serial Monitor, choose command format as “Both NL & CR”and 115200.
4. Send “**Calibration**” to enter Calibration Mode, and you will see “Enter Calibration Mode” directly.



5. Acid Calibration

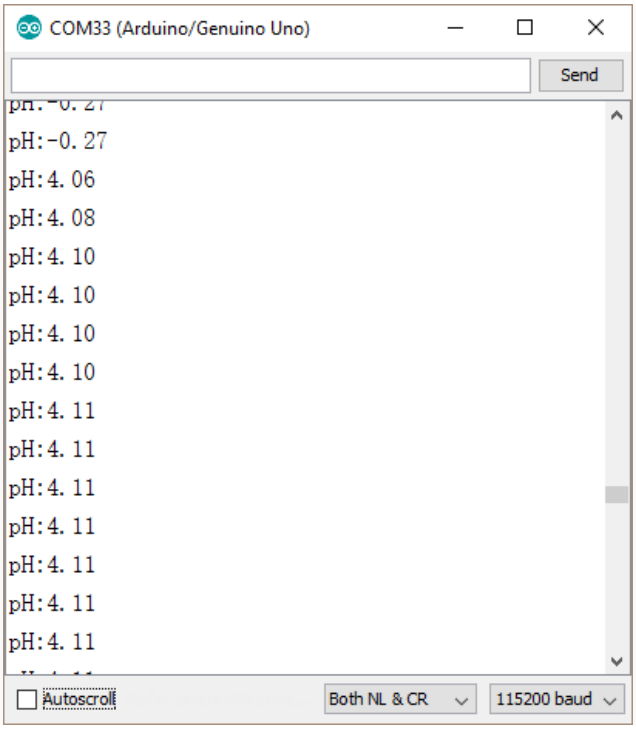
1. Wash your pH probe with pure water (distilled water is best) and dryer it in case of diluting the standard pH solution. Insert it into standard acid solution of pH = 4.0. Wait several seconds till the readings get relative stable.
2. Enter “**acid:4.00**”(no bland space, lower case), and you will get “Acid Calibration Successful” notice. Then go on with Alkali Calibration.

6. Alkali Calibration

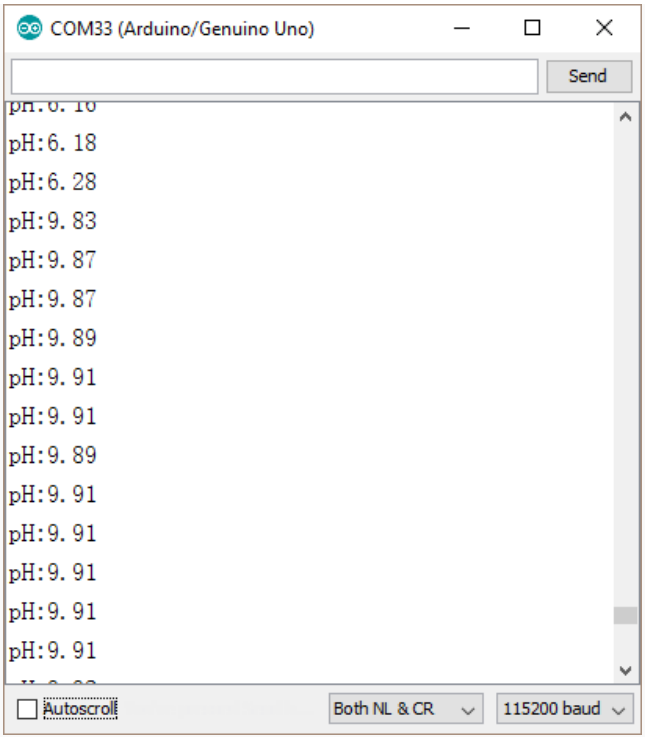
1. Take out the pH probe out of the acid solution, CLEAN it again as you did in last step. After this, insert it into the standard alkali solution with pH = 10 or 9.18. Waiting for the stable readings
2. Enter “**alkali:10.00**”, and you will see “Alkali Calibration Successful”.

7. Enter “exit” to finish calibration. You will see “Calibration Successful,Exit Calibration Mode”.

8. Check if the pH meter was calibrated successfully with the solution pH = 4.00, 9.18, 10.00, if the readings are within the error of 0.1. Congrats!

- 

In Standard acid solution pH = 4.00

- 

In Standard alkali solution pH = 10.00

Sample code: Software Calibration

```
/*************************************************

 This example uses software solution to calibration the ph meter,
not the potentiometer. So it is more easy to use and calibrate.

 This is for SEN0161 and SEN0169.


 Created 2016-8-11

 By youyou from DFrobot <youyou.yu@dfrobot.com>


 GNU Lesser General Public License.

 See <http://www.gnu.org/licenses/> for details.

 All above must be included in any redistribution

 *************************************************/


/***********Notice and Troubleshooting***************

 1.Connection and Diagram can be found here  http://www.dfrobot.co
m/wiki/index.php/PH_meter%28SKU:_SEN0161%29

 2.This code is tested on Arduino Uno.

 *************************************************/


#include <EEPROM.h>

#define EEPROM_write(address, p) {int i = 0; byte *pp = (byte*)&(p
);for(; i < sizeof(p); i++) EEPROM.write(address+i, pp[i]);}

#define EEPROM_read(address, p)  {int i = 0; byte *pp = (byte*)&(p
);for(; i < sizeof(p); i++) pp[i]=EEPROM.read(address+i);}


#define ReceivedBufferLength 20

char receivedBuffer[ReceivedBufferLength+1];   // store the serial
command

byte receivedBufferIndex = 0;
```

```
#define SCOUNT  30           // sum of sample point

int analogBuffer[SCOUNT];    //store the sample voltage

int analogBufferIndex = 0;


#define SlopeValueAddress 0     // (slope of the ph probe)store at the beginning of the EEPROM. The slope is a float number,occupies 4 bytes.

#define InterceptValueAddress (SlopeValueAddress+4)

float slopeValue, interceptValue, averageVoltage;

boolean enterCalibrationFlag = 0;


#define SensorPin A0

#define VREF 5000  //for arduino uno, the ADC reference is the pow er(AVCC), that is 5000mV


void setup()

{

  Serial.begin(115200);

  readCharacteristicValues(); //read the slope and intercept of th e ph probe

}


void loop()

{

  if(serialDataAvailable() > 0)

  {

      byte modeIndex = uartParse();

      phCalibration(modeIndex);    // If the correct calibration c ommand is received, the calibration function should be called.

      EEPROM_read(SlopeValueAddress, slopeValue);     // After cal ibration, the new slope and intercept should be read ,to update cu rrent value.

      EEPROM_read(InterceptValueAddress, interceptValue);

  }
```

```
   static unsigned long sampleTimepoint = millis();
   if(millis()-sampleTimepoint>40U)
   {
     sampleTimepoint = millis();
     analogBuffer[analogBufferIndex] = analogRead(SensorPin)/1024.
0*VREF;    //read the voltage and store into the buffer,every 40ms
     analogBufferIndex++;
     if(analogBufferIndex == SCOUNT)
         analogBufferIndex = 0;
     averageVoltage = getMedianNum(analogBuffer,SCOUNT);   // read
the stable value by the median filtering algorithm
   }


   static unsigned long printTimepoint = millis();
   if(millis()-printTimepoint>1000U)
   {
     printTimepoint = millis();
     if(enterCalibrationFlag)             // in calibration mode,
print the voltage to user, to watch the stability of voltage
     {
       Serial.print("Voltage:");
       Serial.print(averageVoltage);
       Serial.println("mV");
     }else{
     Serial.print("pH:");              // in normal mode, print th
e ph value to user
     Serial.println(averageVoltage/1000.0*slopeValue+interceptValu
e);
     }
   }
}


boolean serialDataAvailable(void)
{
```

```
  char receivedChar;
  static unsigned long receivedTimeOut = millis();
  while (Serial.available()>0)
  {
    if (millis() - receivedTimeOut > 1000U)
    {
      receivedBufferIndex = 0;
      memset(receivedBuffer,0,(ReceivedBufferLength+1));
    }
    receivedTimeOut = millis();
    receivedChar = Serial.read();
    if (receivedChar == '\n' || receivedBufferIndex==ReceivedBuffe
rLength){
                receivedBufferIndex = 0;
                strupr(receivedBuffer);
                return true;
    }
    else{
      receivedBuffer[receivedBufferIndex] = receivedChar;
      receivedBufferIndex++;
    }
  }
  return false;
}


byte uartParse()
{
  byte modeIndex = 0;
  if(strstr(receivedBuffer, "CALIBRATION") != NULL)
      modeIndex = 1;
  else if(strstr(receivedBuffer, "EXIT") != NULL)
      modeIndex = 4;
  else if(strstr(receivedBuffer, "ACID:") != NULL)
```

```
      modeIndex = 2;
  else if(strstr(receivedBuffer, "ALKALI:") != NULL)
      modeIndex = 3;
  return modeIndex;
}


void phCalibration(byte mode)
{
    char *receivedBufferPtr;
    static byte acidCalibrationFinish = 0, alkaliCalibrationFinish
= 0;
    static float acidValue,alkaliValue;
    static float acidVoltage,alkaliVoltage;
    float acidValueTemp,alkaliValueTemp,newSlopeValue,newIntercept
Value;
    switch(mode)
    {
      case 0:
      if(enterCalibrationFlag)
         Serial.println(F("Command Error"));
      break;

      case 1:
      receivedBufferPtr=strstr(receivedBuffer, "CALIBRATION");
      enterCalibrationFlag = 1;
      acidCalibrationFinish = 0;
      alkaliCalibrationFinish = 0;
      Serial.println(F("Enter Calibration Mode"));
      break;

      case 2:
      if(enterCalibrationFlag)
      {
```

```
          receivedBufferPtr=strstr(receivedBuffer, "ACID:");
          receivedBufferPtr+=strlen("ACID:");
          acidValueTemp = strtod(receivedBufferPtr,NULL);
          if((acidValueTemp>3)&&(acidValueTemp<5))        //typica
l ph value of acid standand buffer solution should be 4.00
          {
             acidValue = acidValueTemp;
             acidVoltage = averageVoltage/1000.0;        // mV ->
V
             acidCalibrationFinish = 1;
             Serial.println(F("Acid Calibration Successful"));
           }else {
             acidCalibrationFinish = 0;
             Serial.println(F("Acid Value Error"));
           }
      }
      break;

       case 3:
       if(enterCalibrationFlag)
       {
           receivedBufferPtr=strstr(receivedBuffer, "ALKALI:");
           receivedBufferPtr+=strlen("ALKALI:");
           alkaliValueTemp = strtod(receivedBufferPtr,NULL);
           if((alkaliValueTemp>8)&&(alkaliValueTemp<11))        //
typical ph value of alkali standand buffer solution should be 9.18
or 10.01
           {
                 alkaliValue = alkaliValueTemp;
                 alkaliVoltage = averageVoltage/1000.0;
                 alkaliCalibrationFinish = 1;
                 Serial.println(F("Alkali Calibration Successful")
);
            }else{
```

```
                alkaliCalibrationFinish = 0;
                Serial.println(F("Alkali Value Error"));
            }
        }
        break;

         case 4:
         if(enterCalibrationFlag)
         {
             if(acidCalibrationFinish && alkaliCalibrationFinish)
             {
               newSlopeValue = (acidValue-alkaliValue)/(acidVoltage
- alkaliVoltage);
               EEPROM_write(SlopeValueAddress, newSlopeValue);
               newInterceptValue = acidValue - (slopeValue*acidVolt
age);
               EEPROM_write(InterceptValueAddress, newInterceptValu
e);
               Serial.print(F("Calibration Successful"));
             }
             else Serial.print(F("Calibration Failed"));
             Serial.println(F(",Exit Calibration Mode"));
             acidCalibrationFinish = 0;
             alkaliCalibrationFinish = 0;
             enterCalibrationFlag = 0;
         }
         break;
    }
}

int getMedianNum(int bArray[], int iFilterLen)
{
      int bTab[iFilterLen];
      for (byte i = 0; i<iFilterLen; i++)
```

```
      {
          bTab[i] = bArray[i];
      }
      int i, j, bTemp;
      for (j = 0; j < iFilterLen - 1; j++)
      {
          for (i = 0; i < iFilterLen - j - 1; i++)
          {
            if (bTab[i] > bTab[i + 1])
            {
                bTemp = bTab[i];
                bTab[i] = bTab[i + 1];
                bTab[i + 1] = bTemp;
            }
          }
      }
      if ((iFilterLen & 1) > 0)
        bTemp = bTab[(iFilterLen - 1) / 2];
      else
        bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1]) /
2;
      return bTemp;
}


void readCharacteristicValues()
{
    EEPROM_read(SlopeValueAddress, slopeValue);
    EEPROM_read(InterceptValueAddress, interceptValue);
    if(EEPROM.read(SlopeValueAddress)==0xFF && EEPROM.read(SlopeVa
lueAddress+1)==0xFF && EEPROM.read(SlopeValueAddress+2)==0xFF && E
EPROM.read(SlopeValueAddress+3)==0xFF)
    {
      slopeValue = 3.5;   // If the EEPROM is new, the recommendat
ory slope is 3.5.
```

```
      EEPROM_write(SlopeValueAddress, slopeValue);
    }
    if(EEPROM.read(InterceptValueAddress)==0xFF && EEPROM.read(Int
erceptValueAddress+1)==0xFF && EEPROM.read(InterceptValueAddress+2
)==0xFF && EEPROM.read(InterceptValueAddress+3)==0xFF)
    {
      interceptValue = 0;  // If the EEPROM is new, the recommenda
tory intercept is 0.
      EEPROM_write(InterceptValueAddress, interceptValue);
    }
}
```

Method 2. Hardware Calibration through potentiometer

If you've taken the Method 1. Software Calibration, you can ignore this part.

1. Connect according to the graphic, that is, the pH electrode is connected to the BNC connector on the pH meter board, and then use the connection lines, the pH meter board is connected to the analog port 0 of the Arduino controller. When the Arduino controller gets power, you will see the blue LED on board is on.
2. Upload the sample code to the Arduino controller.
3. Put the pH electrode into the standard solution whose pH value is 7.00, or directly short circuit the input of the BNC connector. Open the serial monitor of the Arduino IDE, you can see the pH value printed to it, and the error does not exceed 0.3. Record the pH value printed, then compared with 7.00, and the difference should be changed into the "Offset" in the sample code. For example, the pH value printed is 6.88, so the difference is 0.12. You should change the # **define Offset 0.00** into # **define Offset 0.12** in the sample code.
4. **Fine adjustment**

- **For Acid solution:** Put the pH electrode into the pH standard solution whose value is 4.00. Then wait about a minute, adjust the Gain Potential device, let the value stabilise at around 4.00. At this time, the acidic calibration has been completed and you can measure the pH value of an acidic solution.
- **For Alkaline solution:** According to the linear characteristics of pH electrode itself, after the above calibration, you can

directly measure the pH value of the **alkaline** solution, but if you want to get a better accuracy, you can recalibrate it with the standard solution, pH = 9.18. Also adjust the gain potential device, let the value stabilise at around 9.18. After this calibration, you can measure the pH value of the alkaline solution.

Sample Code for Hardware Calibration

```
/*
 # This sample code is used to test the pH meter V1.0.
 # Editor : YouYou
 # Ver    : 1.0
 # Product: analog pH meter
 # SKU    : SEN0161
*/
#define SensorPin A0            //pH meter Analog output to Arduin
o Analog Input 0
#define Offset 0.00            //deviation compensate
#define LED 13
#define samplingInterval 20
#define printInterval 800
#define ArrayLenth  40    //times of collection
int pHArray[ArrayLenth];   //Store the average value of the sensor
feedback
int pHArrayIndex=0;
void setup(void)
{
  pinMode(LED,OUTPUT);
  Serial.begin(9600);
  Serial.println("pH meter experiment!");    //Test the serial mon
itor
}
```

```
void loop(void)
{
  static unsigned long samplingTime = millis();
  static unsigned long printTime = millis();
  static float pHValue,voltage;
  if(millis()-samplingTime > samplingInterval)
  {
      pHArray[pHArrayIndex++]=analogRead(SensorPin);
      if(pHArrayIndex==ArrayLenth)pHArrayIndex=0;
      voltage = avergearray(pHArray, ArrayLenth)*5.0/1024;
      pHValue = 3.5*voltage+Offset;
      samplingTime=millis();
  }
  if(millis() - printTime > printInterval)   //Every 800 milliseco
nds, print a numerical, convert the state of the LED indicator
  {
        Serial.print("Voltage:");
        Serial.print(voltage,2);
        Serial.print("    pH value: ");
        Serial.println(pHValue,2);
        digitalWrite(LED,digitalRead(LED)^1);
        printTime=millis();
  }
}
double avergearray(int* arr, int number){
  int i;
  int max,min;
  double avg;
  long amount=0;
  if(number<=0){
    Serial.println("Error number for the array to avraging!/n");
    return 0;
  }
```

```
  if(number<5){   //less than 5, calculated directly statistics
    for(i=0;i<number;i++){
      amount+=arr[i];
    }
    avg = amount/number;
    return avg;
  }else{
    if(arr[0]<arr[1]){
      min = arr[0];max=arr[1];
    }
    else{
      min=arr[1];max=arr[0];
    }
    for(i=2;i<number;i++){
      if(arr[i]<min){
        amount+=min;        //arr<min
        min=arr[i];
      }else {
        if(arr[i]>max){
          amount+=max;    //arr>max
          max=arr[i];
        }else{
          amount+=arr[i]; //min<=arr<=max
        }
      }//if
    }//for
    avg = (double)amount/(number-2);
  }//if
  return avg;
}
```

FAQ

Q1. My PH sensor readings are not correct, what did I miss?

Or the module is defective?

A. 1. Check if the pH sensor circuit board is good? Read on the Forum. or on wiki for the steps. During the transport, there might be crash causing the probe head cracked, please check if the probe is good or not.

2. If you don't use Arduino as the controller, then please check your ADC module that whether it converts the 5V analog input to 1024, if it is 4096(or other byte), please re-determine the equation in the code.

Q2. Big fluctuations in ph meter readings. When I make measurements in a glass, I have correct, stable reading. But when I put it inside the aquarium with the pumping system working, the easurement varies even more than a degree, and it's not stable, if I swicth off the pump the given value doesn´t oscilate anymore.

A. There should be NO working electrical device in the container. Any tiny leakage of electricity will cause the probe working error. Especially, many people bought the EC meter and put it into the same tank for the test, but then the pH meter cannot work well anymore. Please seperate them into different containers, or turning off the EC meter when using the pH meter.

Q3. May I know the Maximum range different if we do not calibrate the pH meter.

A. The maximum range differs from probe, you have to calibrate it before use if the pH probe was kept long.

Q4. I would just like to ask if your pH sensor can be connect to any micro controller aside from arduino. Would it be compatible with a raspberry pi? Thank You!

A. Yes, it can be used on any device as long as it could give 5V power supply and accept 5V analog signal, but as the Rasp pi is only compatible with 3.3V sensor, so an expansion shield is suggested to use with (please make sure which kind of Pi you use)

For any questions and more cool ideas to share, please visit **DFRobot Forum**