

基于物联网的离网光伏发电装置监测系统设计与实现

2022 年全国大学生节能减排社会实践与科技竞赛报告

目录

小组信息	3
1 摘要	4
2 研究背景及意义	5
2.1 课题研究背景	5
2.2 物联网技术背景	7
2.3 课题研究现状	8
2.4 作品概述与意义	9
3 系统方案	10
3.1 系统设计	10
3.2 系统流程	10
4 硬件部分设计	11
4.1 设计方案	11
4.2 各模块介绍	12
5 软件部分设计	20
5.1 参数计算	20
5.1.1 Arduino IDE	20
5.1.2 Code {Language == C++}	20
5.1.3 代码解释	22
5.2 通过 Wi-Fi 上传到 Arduino Cloud	24
5.2.1 Arduino Cloud	24
5.2.2 连接 ESP32 和 Arduino Cloud	24
5.2.3 测试结果	26
5.3 通过 Wi-Fi 或窄带物联网 (NB-IoT) 上传数据至私有云服务器	28
5.3.1 应用场景	28
5.3.2 关键技术	29
5.3.3 WiFi 连接	29
5.3.4 NB-IoT 连接	32

5.3.5	测试结果	34
6	数据处理与分析	35
7	特色与创新	36
8	问题与展望	37
9	参考文献	37



华南理工大学
South China University of Technology

学院: 电力学院

专业: 电气工程及其自动化

成员: 赵知易 祁宇轩 向美云 张凌云

学号: 202030220186 202030220056 202066236038 202030220094

起始日期: 2022 年 4 月 5 日——2022 年 4 月 25 日

1 摘要

离网光伏发电有广泛的市场及应用前景，其对节能减排有着显著的作用。为此我们设计了一款成本低廉、集成在手机大小 PCB 板上的太阳能板监测装置，其通过传感器采集太阳能板的实时电压、电流、温度数据，由 $P = U * I$ 计算出功率，并基于当地电价计算出节省的费用。

当下是一个万物联网的时代，为了普及新能源的使用，将光伏发电装置的各个参数上传云端，实时反馈给用户端很有必要。所以我们在此基础上基于物联网技术将硬件采集到的参数传入云端，便于用户查看。平台还通过一系列算法对采集到的信息进行能耗分析，通过系统设置的阀值判断其是否发生故障，有效地降低离网太阳能板的维护成本，

我们观察到市面上的物联网产品都依赖于阿里云、华为云等私企云，繁琐的登陆流程及环境配置使联网变得困难，固定的前端设计难以满足用户个性化的需求，所以我们创新地搭建了个人 Web 服务器，在 SQL 服务器中提取数据显示在前端界面。实现了用户端的即插即用，零配置的产品特点。

离网型太阳能光伏发电系统主要应用于偏僻山区、无电区、海岛、通讯基站等用电不方便场所，而以往的产品往往仅依赖于 Wi-Fi 进行数据传输，极大限制了监测的应用场景，所以我们在 Wi-Fi 上云的基础上，又引入了 5G 模块，应用低功耗的 NB-IoT 窄带物联网通信技术进行上云。拓宽了应用场景。

关键词：离网光伏发电装置；物联网技术；NB-IoT 窄带物联网通信技术；数据监测；能耗分析；故障检测

2 研究背景及意义

能源是推动世界经济发展的车轮，然而随着时代的发展，我们对于能源的过度消耗已经导致了全球性的环境污染和资源枯竭。如何解决社会发展对能源的巨大需求与有限的资源之间的矛盾成为摆在我们这一代人面前的重要命题。目前，大力发展新能源是时代的大势所趋，而在一众新能源中，太阳能又凭借其分布广泛，储量丰富，用之不竭的特点得到广泛应用。目前并网太阳能发展已趋于稳定和成熟，但离网太阳能的市场仍处于初级阶段，还有一些亟待解决的问题，基于此种情况，我们开展了此课题的研究。

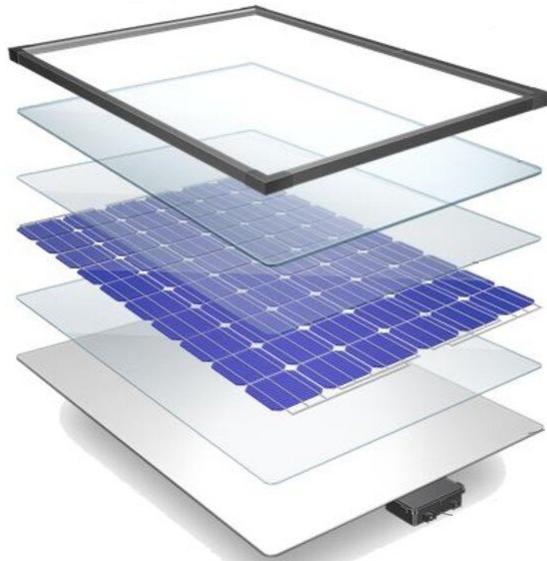


Figure 1: 太阳能板结构

2.1 课题研究背景

2009 年，世界银行集团的点亮非洲项目提出了一个大胆革新的目标：到 2030 年，使 2.5 亿人口通过使用离网太阳能产品，接入可持续的清洁电力。得益于一代人的远见、创新和付出，离网太阳能行业在 2019 年就已实现并超越 2.5 亿使用人口的目标。如今，作为消除能源贫穷战斗中的关键组成部分，离网太阳能行业在全球范围内已被欣然接受。实践证明，离网太阳能已经成为在全球农村社区提供可负担得起的现代电力服务的可靠路径，并成为电网电力不稳定地区的重要补充。通过过去十年的蓬勃发展，离网太阳能行业已成为一个充满活力的年营业额达 17.5 亿美元的市场，保持着稳定的增长态势，在全球范围内为超过 4.2 亿用户提供着能源服务。随着行业走向成熟，企业越来越关注财务健康。在既有市场逐渐饱和的情况下，企业也逐渐转向开发新的地区市场和服务水平低下的市场。全球离网太阳能拥有巨大的潜在市场，全球有 8.4 亿人口尚未接入电力，超过 10 亿人口连接的电网供电不稳定，超过 7000 万农民利用离网太阳能工具或设备进行生产活动。多个迹象表明该行业的经济成熟度不断提高，如债权投资的增加和融资数额的增大等。

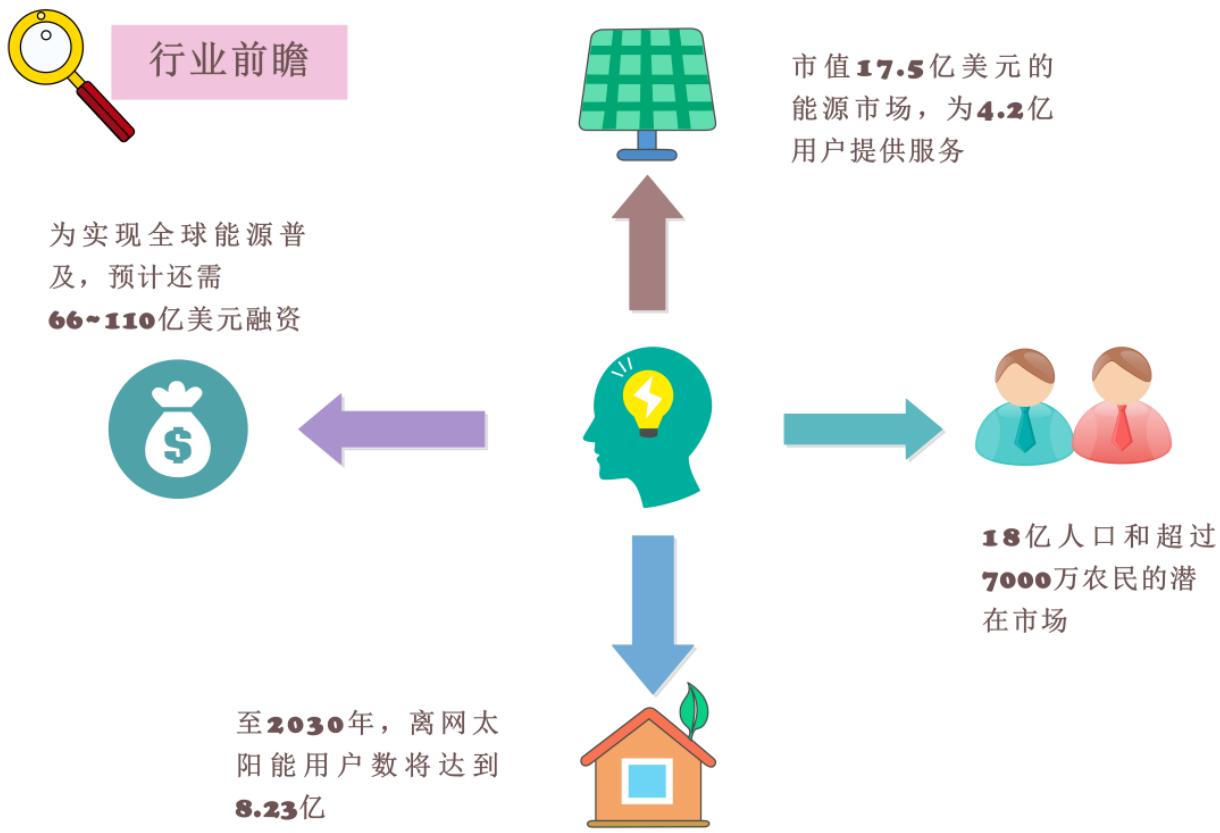


Figure 2: 行业前瞻

离网光伏发电装置主要有以下常见应用场景：

1. 用户太阳能电源：小型电源 10-100W 不等，用于边远无电地区如高原、海岛、牧区、边防哨所等军民生活用电，如照明、电视、收录机等。光伏水泵，解决无电地区的深水井饮用、灌溉。家庭灯具电源，如庭院灯、路灯、手提灯、野营灯、登山灯、垂钓灯、黑光灯、割胶灯、节能灯等。
2. 交通领域如航标灯、交通/铁路信号灯、交通警示/标志灯、宇翔路灯、高空障碍灯、高速公路/铁路无线电话亭、无人值守道班供电等。
3. 通讯/通信领域：太阳能无人值守微波中继站、光缆维护站、广播/通讯/寻呼电源系统；农村载波电话光伏系统、小型通信机、士兵 GPS 供电等。
4. 石油、海洋、气象领域：石油管道和水库闸门阴极保护太阳能电源系统、石油钻井平台生活及应急电源、海洋检测设备、气象/水文观测设备等。
5. 光伏电站：10KW-50MW 独立光伏电站、风光（柴）互补电站、各种大型停车场充电站等。其他领域包括还太阳能制氢加燃料电池的再生发电系统，海水淡化设备供电等。
6. 其他领域还包括太阳能制氢加燃料电池的再生发电系统，海水淡化设备供电等。

将离网太阳能产品应用于生产活动领域是离网太阳能行业的一个新兴机会，这块市场目前尚处于初级阶段，尚需学习和发展。生产用太阳能领域的产品范围十分广泛，如太阳能水泵、冷藏冷冻设备和太阳能磨粉设备等公用设施服务都是传统离网太阳能市场的自然扩展应用，可同时服务于家用场景和小微企业，两个市场都拥有着巨大的潜在机会。



Figure 3: 光伏发电应用场景

2.2 物联网技术背景

物联网是指通过RFID、感应器等信息传感设备及技术，按约定的协议，把物品与互联网连接起来，进行信息交换和通信，以实现智能化识别、定位、跟踪、监测和管理的一种网络概念。物联网可分为四层组成架构，分别为感知层、传输层、平台层和应用层。感知层实现物联网全面感知的核心能力，网络层主要以广泛覆盖的移动通信网络作为基础设施，形成系统感知的网络。应用层提供丰富的应用，满足行业信息化需求。

如今有很多商业模式正在利用物联网，比如智能设备、运输、农业等，并已经产生了相对广泛的连接平台。无所不在的物联网通信时代即将到来，物联网的发展将催生一大批新兴产业，提高生产效率和企业竞争力，推动人与自然的协调发展。在这样的技术背景下，学习、研究并将物联网技术应用到实际中，就显得尤为必要。

物联网在万物互联的大趋势下扮演着重要角色，其中，NB-IoT、5G通信模组是获取“物”、大数

据最关键、最核心的基础通信单元，是新基建领域的核心通信器件，正加速渗透到生产、消费、安防和社会管理等各层面、各领域。从宏观层面来看，NB-IoT 技术其实是 5G 技术的“敲门砖”，而 NB-IoT 技术也可依托 5G 技术实现升级以长远发展，二者可实现协同发展。NB-IoT 聚焦于低功耗广覆盖物联网市场，是一种可在全球范围内广泛应用的新兴技术。其具有覆盖广、连接多、速率低、成本低、功耗低、架构优等特点。因为 NB-IoT 自身具备此类优势，故其可以广泛应用于多种垂直行业，如远程抄表、资产跟踪、智能停车、智慧农业等。

2.3 课题研究现状

2021 年 9 月，工信部等八部门印发《物联网新型基础设施建设三年行动计划（2021-2023 年）》，明确到 2023 年底，在国内主要城市初步建成物联网新型基础设施，社会现代化治理、产业数字化转型和民生消费升级的基础更加稳固。目前，从国际上看，许多发达地区和国家都十分重视对物联网的研究，其研究开发工作和实验性应用已经得到了广泛的进行，各国都陆续提出了物联网的发展战略，如美国的“智慧地球”，欧盟的“欧盟物联网行动计划”，日韩的“U 战略”等，发展物联网毫无疑问是新的经济增长点。

随着 5G 时代的到来，我们比以往更加渴望建立一个全移动、万物互联、充满想象的智慧社会。5G 依托全新的网络架构，具有高速率、低延时、高可靠性、大宽带等优势，不仅满足“人与人”之间的多元化通信需求，还将逐步渗透至物联网、工业自动化等领域。我国政府将 5G 纳入国家战略，视为实施国家创新战略的重点之一。目前 5G 在中国具有良好的前景，而未来有望持续保持这种优势，进一步扩大市场。



Figure 4: 5G 技术

目前，许多 5G 物联网相关技术应用仍处于研究测试阶段，新能源问题也是物联网应用的强化研究方向。在光伏发电领域的实际应用时，发电系统通常都包括数量巨大且位置分散的电池板。同时由于太阳能电池板的维护保养与更新工作都较频繁，目前尚未有对分散的太阳能电池板进行统一监测与管理的系统，这对太阳能发电体系的全面推进会产生一定负面影响。而其他领域的一些监测系统，如：光缆监

测系统、矿井安全检测系统、水污染监测系统等大都采用有线连接或 WIFI 连接的方式进行监控，虽然能够适应太阳能发电规模大且分散的特点，但是成本普遍偏高且不便于用户查看和使用，为了实现低能耗、低成本、高效率的监测，我们把物联网引入到该体系中，提出了基于物联网的光伏发电监测系统。

2.4 作品概述与意义

随着科技的发展，物联网的概念已深入人心，其相关技术也已经在智能家居、智慧物流等方面得到发展和突破，逐渐融入人们的生活。但是由于太阳能发电的局限性，物联网技术在该行业运用相对较少，也缺乏健全的可视化的运维体系。

同时，虽然离网光伏行业具有良好的前景和市场，但在实际发展中却并不十分顺利，那是因为该行业还存在着一些亟待解决的问题。太阳能电池的发电效率和使用寿命都容易受到种种内外因素影响，增加了维修成本，在使用过程中给用户带来种种不便。由于过去的市场竞争始终围绕着价格展开，随着用户对离网太阳能产品日益熟悉，质量成为了市场竞争中越来越重要的因素。经过质量认证的产品开始逐步取代之前便宜、低质量的产品。如何延长产品的使用期限，提高发电效率，减少发电装置故障的产生，降低运营和维修成本，成了目前行业发展的方向。

以下是我们结合生活实际总结的几种常见的导致光伏发电效率降低或产生故障的情况：

1. 受环境因素影响导致效率降低。光伏发电受环境的影响很大，这也是它目前最明显的缺陷之一。除了人为的损坏和遮挡之外，空气中的颗粒物如灰尘等落在太阳能电池组件的表面都会在不同程度上阻挡光线照射，降低发电效率，甚至有可能导致电池板损坏。
2. 太阳能电池板损坏，其最常见的是太阳能电池隐裂。这主要是由于出产工艺和人工操作失误带来的损坏。虽然这些损坏只会在电池板上留下很小的裂痕，肉眼几乎不可见，但它们带来的损失却是不可忽视的。实验证明，太阳能板上 50 微米到 4 公厘不等的裂痕，会造成 0.9%~42.8% 的能源损耗，若没有定期检查，任由损坏的电池工作的话，甚至可能引起短路或失火。
3. 蓄电池温度升高故障。当环境温度过高，电池柜通风不良，浮充电压过高或电流过大，电池内部短路等种种情况均会导致蓄电池温度升高而引发故障。如果在充电过程中监测到温度不正常升高，并相应的做出调整，如：降低环境温度，改善通风条件，纠正充电系统，更换短路电池等，便有望及时止损，维护设备的正常运行。
4. 光伏组件 PID 效应，又称电势诱导衰减效应，是电池组件的封装材料和其上表面及下表面的材料，电池片与其接地金属边框之间的高电压作用下出现离子迁移，而造成组件性能衰减的现象。当出现此效应时，光伏组件性能急剧衰减，功率大幅度降低。

基于此类情况，为了推动能源的最大化使用，提高安全性能，优化系统的运营维护与管理，提升用户的使用感，我们可以通过实时监测和收集太阳能板的电压、电流、功率和温度等各项指标，再利用物联网技术传递到云端进行信息的呈现、分析和比较，给用户提供一个可视化的监测平台。利用平台提供的信息，用户可以根据实际情况对光伏发电装置进行查看、清理和检修，从而大大减少运营成本。

因此，为了适应国内新能源产业蓬勃发展的趋势，解决现有的问题，我们希望将物联网技术应用到离网光伏发电系统中，利用物联网技术搭建智慧光伏发电能源管控可视化系统，根据光伏发电现状、问题和能源管理特点，充分考虑各能耗管理的整合，设计科学高效，可实施性强，便于运维管理的解决方案。为离网光伏发电系统提供一个全面实时、可感可知的数据监测决策分析平台。

3 系统方案

3.1 系统设计

通过电路设计对太阳能的电压、电流、温度三个参量进行测量，并将数据传入 ESP32 开发板，开发板通过 Wi-Fi 传输数据到 Arduino Cloud。

通过 Wi-Fi 基于 socket 通信传入 PC，或通过构建于蜂窝网络的窄带物联网（Narrow Band Internet of Things, NB-IoT）传输数据到流转服务器，进行数据解析后传到 SQL Server，最后设定协议传入私有云服务器，在个性化的前端里显示数据反馈给使用者。

3.2 系统流程

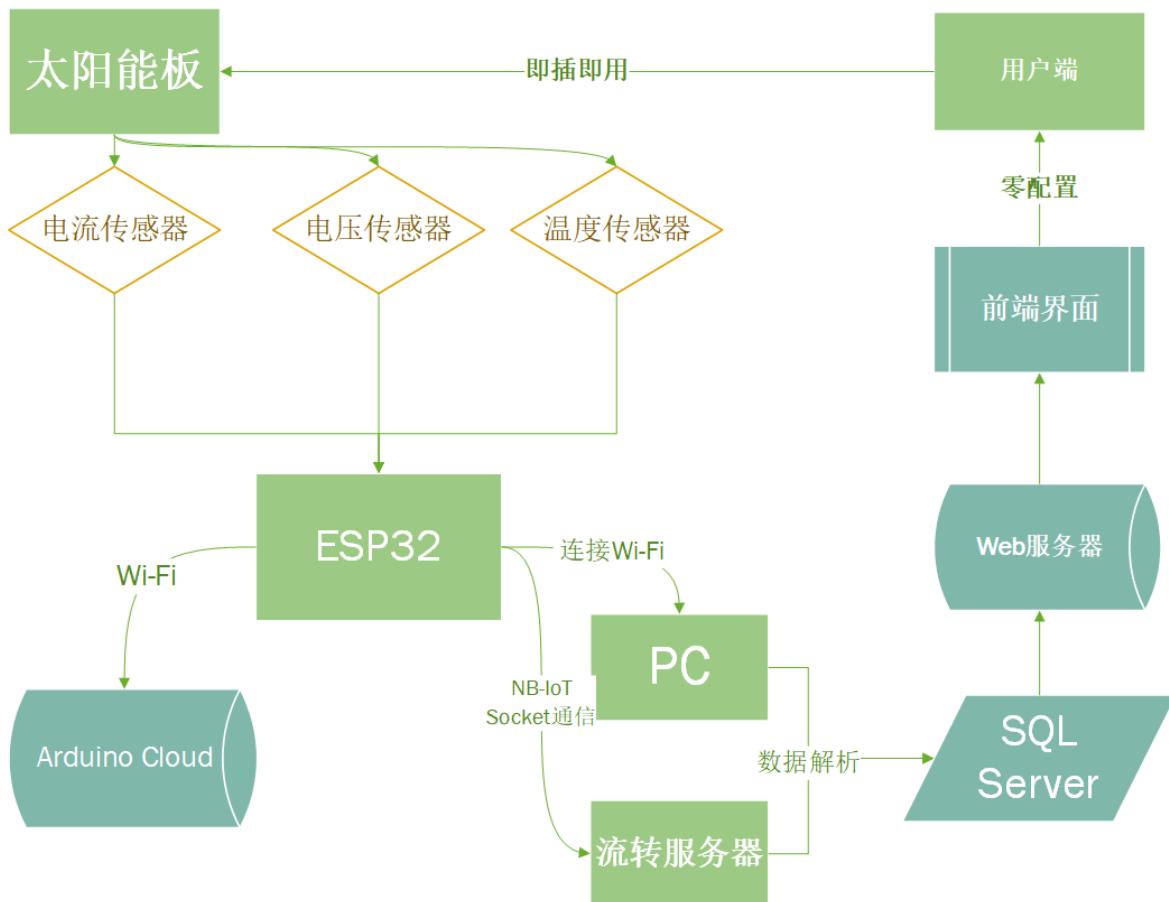


Figure 5: 系统流程图

4 硬件部分设计

4.1 设计方案

在本作品的设计中分别使用分压器网络和 ACS723 电流传感器来采集太阳能电池板的电压和电流数据。使用 DS18B20 温度传感器来感知环境温度。所有来自传感器的原始数据由 ESP32 板处理，并进行必要的功率、能量等的运算。经过处理的数据通过 NB-IoT BC26 5G 模块板上传到云端，实现对太阳能板发电情况的远程监控。

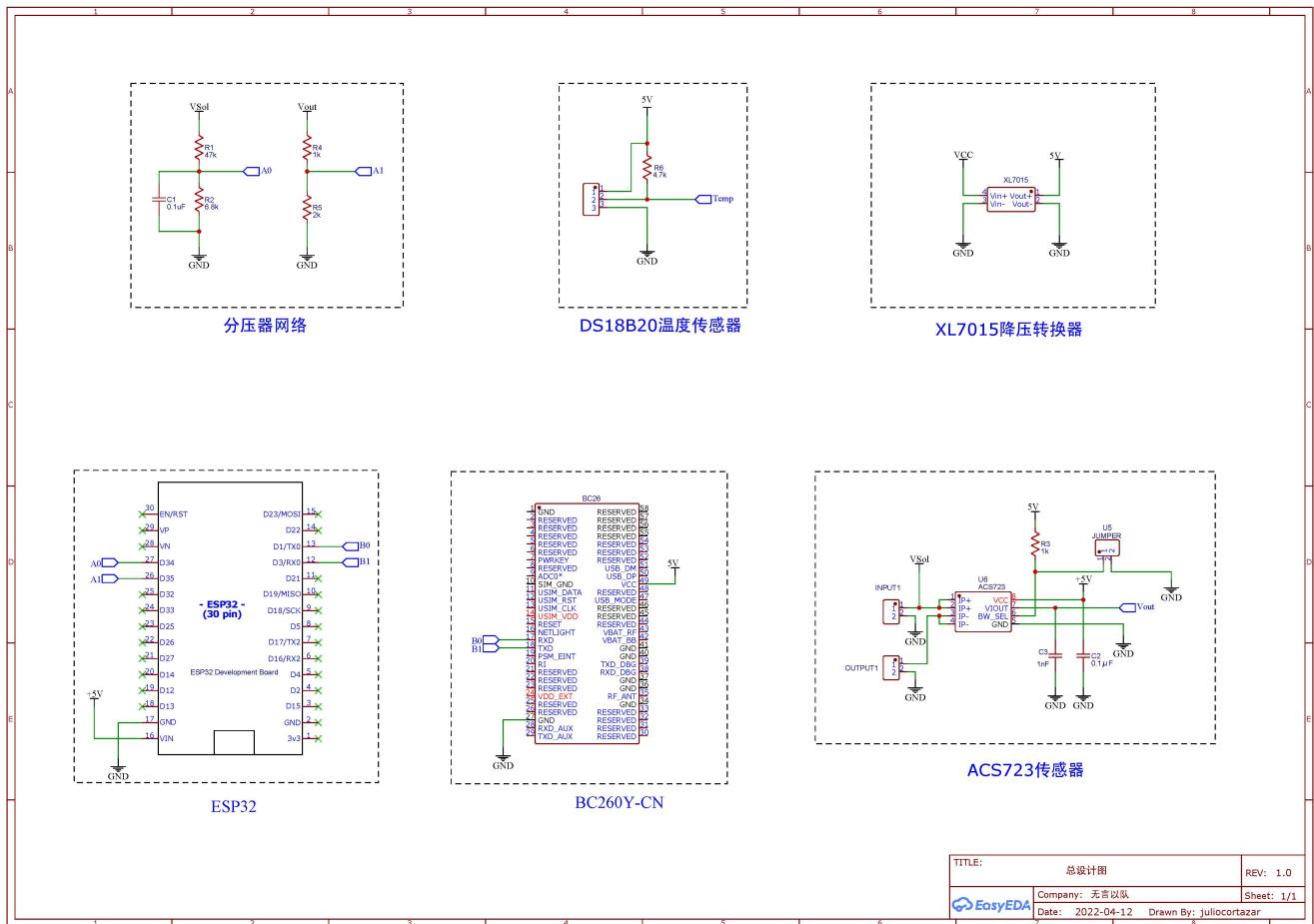


Figure 6: 总设计图

4.2 各模块介绍

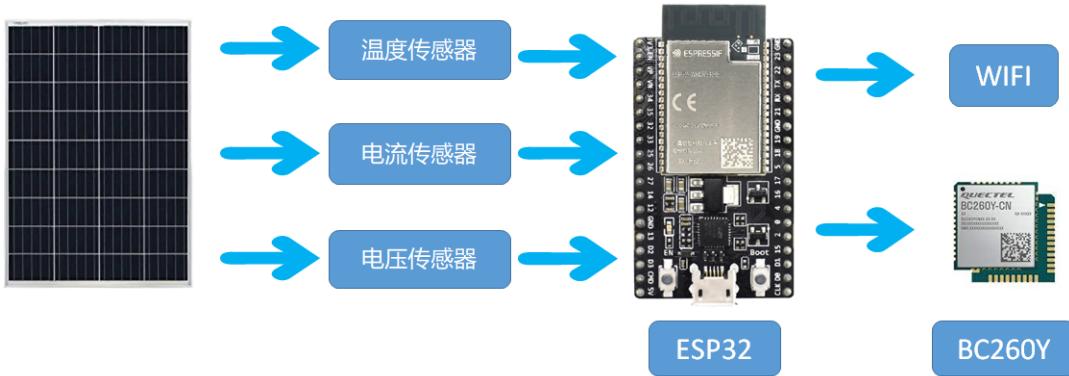


Figure 7: 硬件设计

1. ESP32 Board

ESP32 是低成本，低功耗的单片机微控制器，集成了 Wi-Fi 和双模蓝牙，适用于多样的物联网应用。ESP32 系列采用 Tensilica Xtensa LX6 微处理器，由内置天线开关、RF balun、功率放大器、低噪声接收放大器、滤波器和电源管理模块高度集成。ESP32 能够在工业环境中可靠运行，工作温度范围为 -40°C 至 $+125^{\circ}\text{C}$ 。ESP32 由先进的校准电路提供支持，可以动态消除外部电路缺陷并适应外部条件的变化。ESP32 专为移动设备、可穿戴电子产品和物联网应用而设计，结合多种类型的专有软件实现了超低功耗，还包括最先进的功能，例如细粒度时钟门控、各种功耗模式和动态功耗缩放。在整个系统中，ESP32 接受来自 ACS723 传感器、DS18B20 温度传感器和分压网络监测到的数据，完成初步处理后将数据传输给 5G 模块。

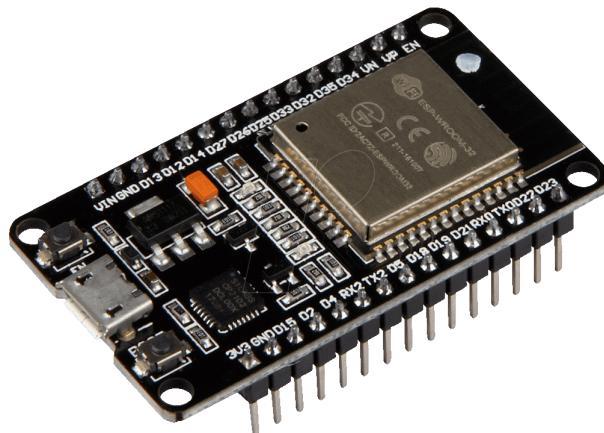


Figure 8: ESP32 板实物图

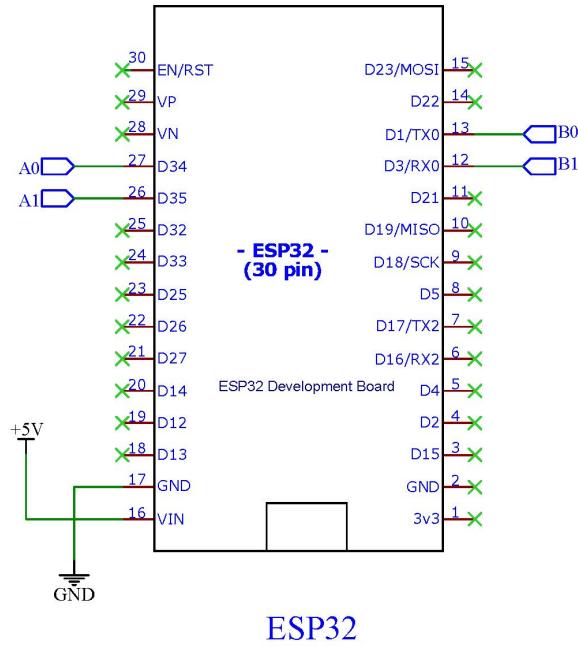


Figure 9: ESP32 板电路图

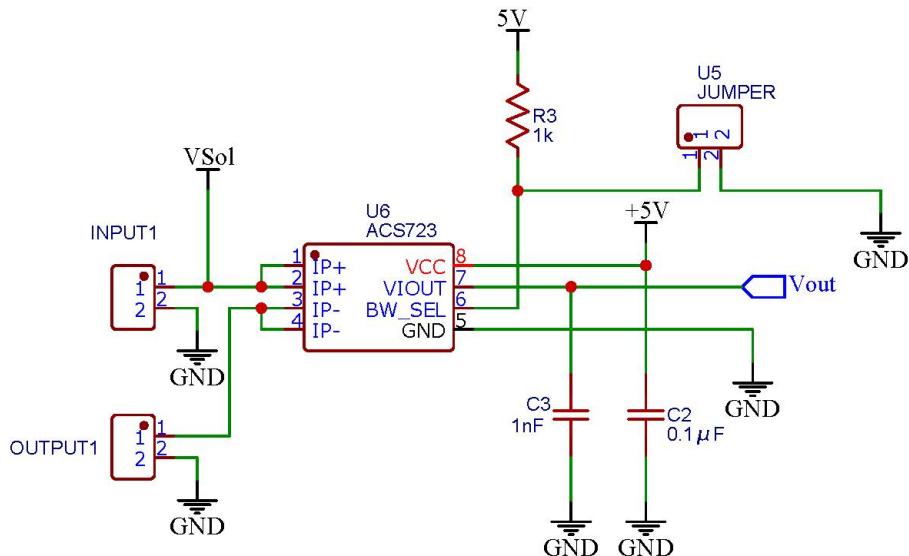
2. ACS723 传感器

ACS723 电流传感器是一款便捷的小型电流传感器，可用于中低电流传感。ACS723 传感器使用霍尔效应传感器输出相对于 IP + 和 IP - 引脚电流的电压。霍尔效应传感器的优点是测量和感应的电路是完全电隔离的。这意味着，虽然普通的 Arduino 是运行在 5V，感应电路可以在更高的直流或交流电压下运行。ACS723 电流传感器 IC 是工业、商业和通信系统交直流传感的精确经济的解决方案。典型应用包括电动机控制、载荷检测和管理、开关式电源和过电流故障保护。该器件具有精确的、低偏移线性霍尔传感器电路，并在晶片表面附近设有铜传导通路。通过该铜传导通路的应用电流能够生成可被集成霍尔 IC 感应并转化为成比例电压的磁场。通过磁场与霍尔传感器的靠近来优化器件精确度。

- 测量直流和交流电流范围: 10mA 5A
- 测量和感应电路的完全电隔离
- 基本灵敏度: 200mV/A
- 传导路径内部电阻阻值: 0.65mΩ; 0.85mΩ
- 在本系统中，ACS723 通过霍尔效应测量来自太阳能光伏发电板的电流，ACS723 的输出通过 R4 和 R5 组成的分压器网络进行降压。



Figure 10: ACS723 传感器实物图



ACS723 传感器

Figure 11: ACS723 传感器电路图

3. DS18B20 数字温度传感器 DS18B20 是一款常用的高精度的单总线数字温度测量芯片。具有体积

小，硬件开销低，抗干扰能力强，精度高的特点。传感器参数如下：

- (a) 测温范围为-55°C 到 +125°C，在-10°C 到 +85°C 范围内误差为 ±0.4°
- (b) 返回 16 位二进制温度数值
- (c) 主机和从机通信使用单总线，即使用单线进行数据的发送和接收
- (d) 在使用中不需要任何外围元件，独立芯片即可完成工作
- (e) 掉电保护功能 DS18B20 内部含有 EEPROM，通过配置寄存器可以设定数字转换精度和报警温度，在系统掉电以后，它仍可保存分辨率及报警温度的设定值
- (f) 每个 DS18B20 都有独立唯一的 64 位-ID，此特性决定了它可以将任意多的 DS18B20 挂载到一根总线上，通过 ROM 搜索读取相应 DS18B20 的温度值
- (g) 宽电压供电，电压 2.5V 5.5V
- (h) DS18B20 返回的 16 位二进制数代表此刻探测的温度值，其高五位代表正负。如果高五位全部为 1，则代表返回的温度值为负值。如果高五位全部为 0，则代表返回的温度值为正值。后面的 11 位数据代表温度的绝对值，将其转换为十进制数值之后，再乘以 0.0625 即可获得此时的温度值

在本系统中，DS18B20 温度传感器通过 3 个引脚螺丝端子连接到 PCB 板。安装 onewire 库实现二者的连接。

连接方式：

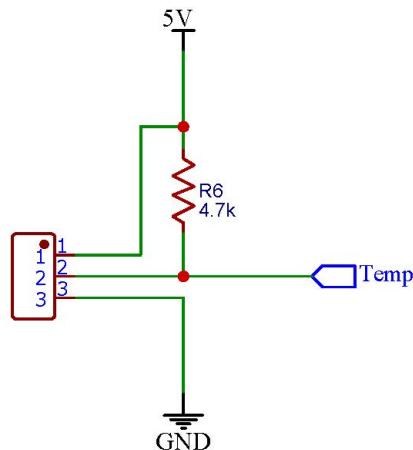
红线-> Vcc

黄线 -> 数据

黑线 -> GND



Figure 12: DS18B20 实物图



DS18B20温度传感器

Figure 13: DS18B20 电路图

4. XL7015 降压转换器

XL7015 DC-DC 降压模块是一款高效、高压降压 DC-DC 转换器，可提供高达 0.8A 的输出电流能力、低纹波、出色的线路调节和负载调节，输入电压最高可达 80V，远超 2596 模块。芯片内置多种保护机制。可应用于电动汽车控制器。具有优良的线性度和负载调整率。

降压器主要参数：

- (a) 输入电压：5V 80V
- (b) 输出电压 5V 20V
- (c) 最小压降 1V
- (d) 最大开关电流 0.8A
- (e) 推荐输出功率小于 7W
- (f) 内置热关断保护、输出短路保护、过流保护
- (g) 尺寸：16mm*44mm；

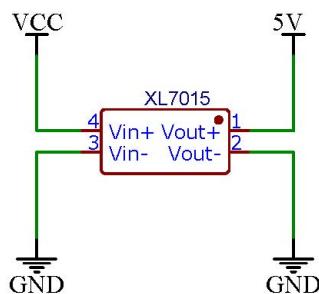
接线：

- (a) VIN：输入电压（5V-80V）
- (b) GND：输入电压接地端
- (c) VOUT：输出电压端（5V-20V）
- (d) GND：接地端的输出电压

在本系统中，使用 XL7015 来进行降压。



Figure 14: XL7015 实物图



XL7015降压转换器

Figure 15: XL7015 电路图

5. OLED Display

6. NB-IoT BC26 board

BC26 board 是一款超高灵敏度、低功耗、多频段 LTE Cat NB1/Cat NB2* 无线通信模块。BC26 在封装设计上兼容移远通信 GSM/GPRS 系列 M26 模块以及 NB-IoT 系列 BC28/BC25/BC260Y-CN 模块，方便客户快速、灵活的进行产品设计和升级。BC26 提供丰富的外部接口和协议栈，同时可支持中国移动 OneNET/Andlink、中国电信 IoT/AEP 以及私人云服务器 IoT 等物联网云平台。支持多频段及丰富外部接口、内嵌网络服务协议栈，应用便利。凭借紧凑的尺寸、超低功耗和超宽工作温度范围，BC26 成为 IoT 应用领域的理想选择，常被用于烟感、无线抄表、共享单车、智能停车、智慧城市、安防、资产追踪、智能家电、可穿戴设备、农业和环境监测以及其它诸多行业，以提供完善的短信和数据传输服务。



Figure 16: BC26 实物图及接口

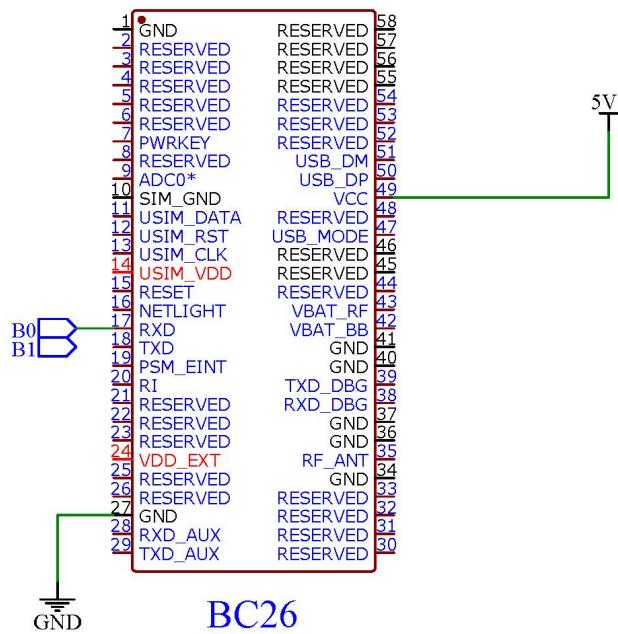


Figure 17: BC26 电路图

其主要参数如下：

- (a) 尺寸：17.7 mm × 15.8 mm × 2.0 mm
- (b) 支持供电范围：2.1~3.63 V
- (c) 支持协议：中国移动 OneNET/Andlink、中国电信 IoT/ AEP 以及私人云服务器 IoT 等

BC26 采用更易于焊接的 LCC 封装，可通过标准 SMT 设备实现模块的快速生产，为客户提供可靠的连接方式，并满足复杂环境下的应用需求。

BC26 模块引出电源接口，PWR 控制引脚，主串口与 DBG 口。只需外接 MCU 板即可调试。

输入电压电流: 5V, 1A

BC26 支持的协议极为丰富，支持 UDP、TCP、LWM2M、MQTT 等协议。STM32G070 的开发板底板可以直接嵌入设备使用。板载温湿度传感器，USB TTL 串口。可以一键调试下载。STM32G0 是 ST 主推的 M0+ 单片机。

5 软件部分设计

5.1 参数计算

5.1.1 Arduino IDE

Arduino 集成开发环境包含一个用于编写代码的文本编辑器、消息区、文本控制台、带有常用功能按钮的工具栏和一系列菜单。它连接到 Arduino 硬件以上传程序并与之通信。Arduino 作一个完全开源、开放的平台，除了官方在 Arduino IDE 中集成了一些自己的一些开板板外，还允许用户根据需要安装或添加第三方的其它非 AVR 单片机内核的开发板，如 ESP32。

5.1.2 Code {Language == C++}

```
1 //————— Loop Function —————
2
3
4
5 void loop()
6 {
7
8     // read voltage and current
9     float voltage = abs( return_voltage_value(INPUT_VOLTAGE_SENSE_PIN) ) ;
10    float current = abs( return_current_value(INPUT_CURRENT_SENSE_PIN) ) ;
11
12    // read temperature from DS18B20
13    sensors.requestTemperatures(); // get temperatures
14    tempC = sensors.getTempCByIndex(0);
15    //tempF = sensors.getTempFByIndex(0);
16
17    // Calculate power and energy
18    power = current * voltage ; // calculate power in Watt
19    last_time = current_time;
20    current_time = millis();
21    energy = energy + power *(( current_time -last_time) /3600000.0) ; //
22    // calculate power in Watt-Hour // 1 Hour = 60mins x 60 Secs x 1000 Milli
23    // Secs
24
25    saving = 6.5 * ( energy /1000 ) ; // 6.5 is cost per kWh // used just for
example
```

```

26 // ===== Display Data on Serial Monitor =====
27
28 Serial.print( "Voltage: " );
29 Serial.println(voltage);
30 Serial.print( "Current: " );
31 Serial.println(current);
32 Serial.print( "Power: " );
33 Serial.println(power);
34 Serial.print( "Energy: " );
35 Serial.println(energy);
36 Serial.print( "Temp: " );
37 Serial.println(tempC);
38 Serial.println(voltage);
39 Serial.println( "Saving: " );
40 Serial.println(saving);
41 delay(1000);
42 }
43
44 //===== Function to Calculate Solar Panel Voltage =====
45
46 double return_voltage_value(int pin_no)
47 {
48     double tmp = 0;
49     double ADCVoltage = 0;
50     double inputVoltage = 0;
51     double avg = 0;
52     for (int i = 0; i < 100; i++)
53     {
54         tmp = tmp + analogRead(pin_no);
55     }
56     avg = tmp / 100;
57     ADCVoltage = ((avg * 3.3) / (4095)) + 0.184 ; // 0.184 is offset adjust
58     by heat and try
59     inputVoltage = ADCVoltage * VOLTAGE_SCALE;
60     return inputVoltage;
61 }
62
63 //===== Function to Calculate Solar Panel Current =====

```

```

64
65 double return_current_value(int pin_no)
66 {
67     double tmp = 0;
68     double avg = 0;
69     double ADCVoltage = 0;
70     double Amps = 0;
71     for (int z = 0; z < 150; z++)
72     {
73         tmp = tmp + analogRead(pin_no);
74     }
75     avg = tmp / 150;
76     ADCVoltage = ((avg*3331) / 4095); // Gets you mV
77     Amps = ((ADCVoltage * CURRENT_SCALE - ACSoffset) / mVperAmp); // 1.5 is
        the scaling for voltage divider
78     return Amps;
79 }
```

5.1.3 代码解释

1. 读取电压

通过 analogRead 函数从 pinno 上读取一个 12bit 的电压值，也是来自于 0v-3.3v 的真实电压到一个 0-4095 整数值的映射。而后需要对这个读取的数据处理，首先重复读取了 100 遍，求出平均值，然后通过 $ADCVoltage = ((avg * 3.3) / (4095)) + 0.184$ 转换得到 pinno 口上的真实电压。由于该端口的电压只是输入电压的分压，再乘上 VOLTAGE_SCALE 得到实际输入的电压。

```

1 double return_voltage_value(int pin_no)
2 {
3     double tmp = 0;
4     double ADCVoltage = 0;
5     double inputVoltage = 0;
6     double avg = 0;
7     for (int i = 0; i < 100; i++)
8     {
9         tmp = tmp + analogRead(pin_no);
10    }
11    avg = tmp / 100;
12    ADCVoltage = ((avg * 3.3) / (4095)) + 0.184 ; // 0.184 is offset
        adjust by heat and try
13    inputVoltage = ADCVoltage * VOLTAGE_SCALE;
```

```

14     return inputVoltage;
15 }
```

2. 读取电流 为了减少测量电路对原电路的影响和方便电流测量，使用了 ACS723 传感器，其作用是将通过对电流转化为电压。后面的操作与读取电压类似，通过 analogRead 函数重复读取 pinno 上的电压值 150 次，取平均值。由于 ACS723 传感器输出的电压值较小，使用 $ADCVoltage = (avg * 3331) / 4095$ 得到 pinon 上的真实电压，单位是 mV。再通过 $Amps = (ADCVoltage * CURRENT_SCALE - ACSOffset) / mVperAmp$ 得到流过 ACS723 传感器的电流。

```

1 double return_current_value(int pin_no)
2 {
3     double tmp = 0;
4     double avg = 0;
5     double ADCVoltage = 0;
6     double Amps = 0;
7     for (int z = 0; z < 150; z++)
8     {
9         tmp = tmp + analogRead(pin_no);
10    }
11    avg = tmp / 150;
12    ADCVoltage = ((avg * 3331) / 4095); // Gets you mV
13    Amps = ((ADCVoltage * CURRENT_SCALE - ACSOffset) / mVperAmp); //
14    // 1.5 is the scaling for voltage divider
15    return Amps;
}
```

3. 读取温度 温度检测使用了 DS18B20 数字温度传感器，这个传感器能输出数字信号，在这里使用了 DallasTemperature.h 和 OneWire.h 两个库。能够通过库提供的接口，读取传感器中的温度值。

```

1 sensors.requestTemperatures(); // get temperatures
2 tempC = sensors.getTempCByIndex(0);
```

5.2 通过 Wi-Fi 上传到 Arduino Cloud

5.2.1 Arduino Cloud

Arduino IoT Cloud 是一款应用程序，帮助制造商以快速、简单和安全的方式构建连接对象。您可以将多个设备相互连接，并允许它们交换实时数据。您还可以使用简单的用户界面从任何地方监视它们。

5.2.2 连接 ESP32 和 Arduino Cloud

- 将电路板设置为第三方设备

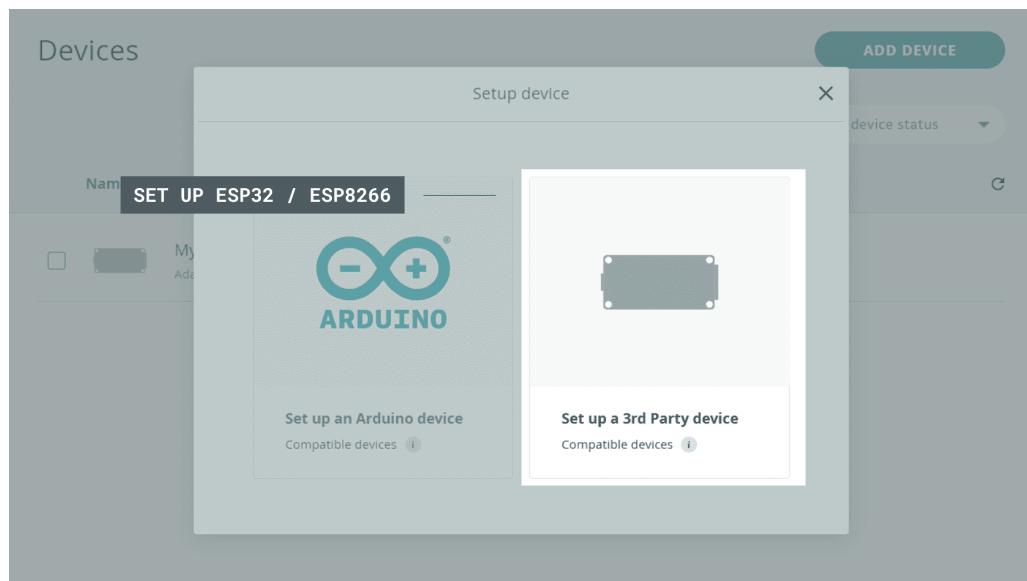


Figure 18: 设置型号

- 选择了正确的设备类型和型号，生成设备 ID 和密钥

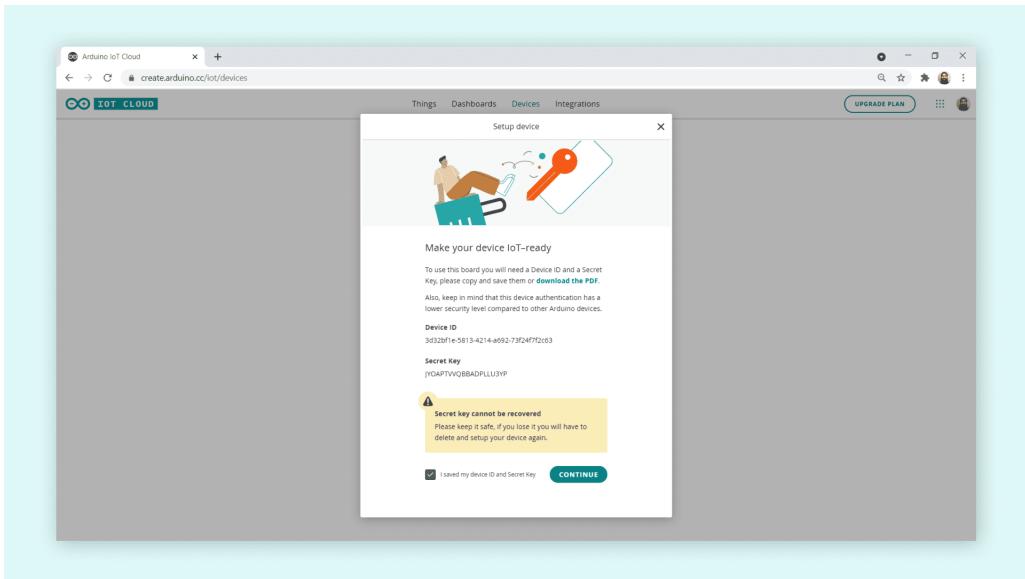


Figure 19: 设备 ID 和密钥

- 添加 Wi-Fi 网络名称（SSID）和密码。还要确保正确添加生成的密钥。

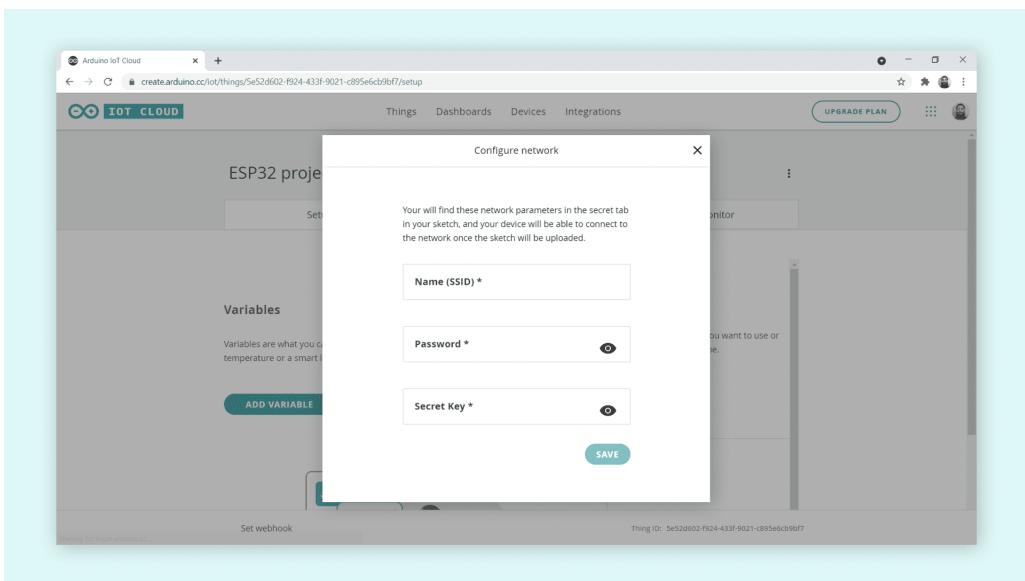


Figure 20: 设置 Wi-Fi 网络 SSID 和密码

- 生成监测变量及可视化仪表板 (Dashboard)

Variables		
Name ↓	Last Value	Last Update
<input type="checkbox"/> Current CloudElectricCurrent current;	9.822	19 Apr 2022 22:21:11
<input type="checkbox"/> Energy CloudEnergy energy;	0.676	19 Apr 2022 22:21:11
<input type="checkbox"/> Power CloudPower power;	16.685	19 Apr 2022 22:21:11
<input type="checkbox"/> Saving float saving;	0.004	19 Apr 2022 22:21:11
<input type="checkbox"/> Temperature CloudTemperatureSensor temperature;	30.563	19 Apr 2022 22:21:11

Figure 21: 变量

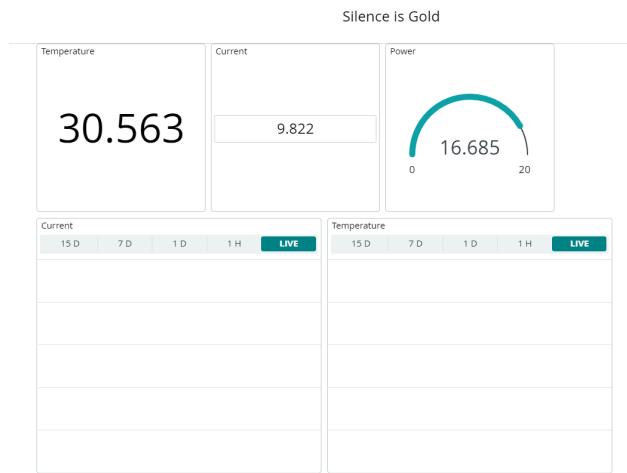


Figure 22: 仪表板

- 添加 thingProperties.h 头文件 (`#include "thingProperties.h"`)
- 在 int main() 函数添加 `ArduinoCloud.begin(ArduinoIoTPreferredConnection);`
- 对 ESP 再次烧录

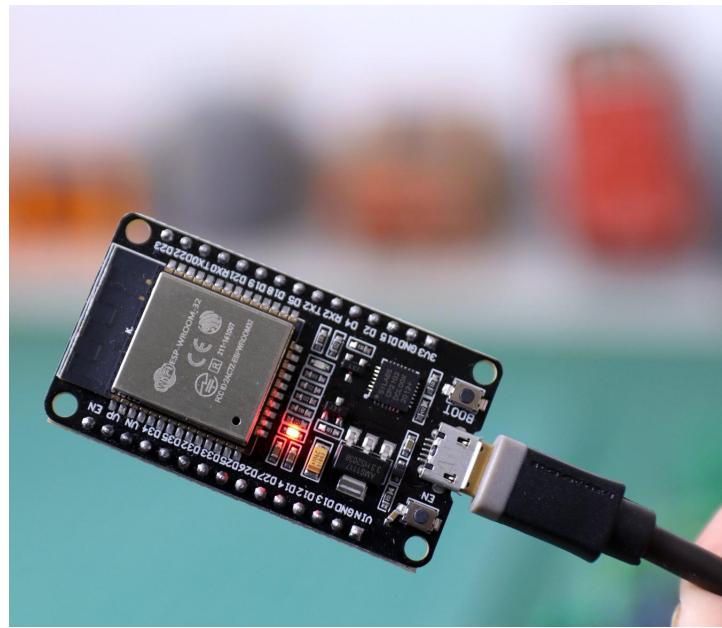


Figure 23: 烧录

5.2.3 测试结果

- 打开串口监视器

```

COM3
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:10124
load:0x40080400,len:5856
entry 0x400806a8
***** Arduino IoT Cloud - configuration info *****
Device ID: 4264e587-832d-4c32-baad-53f26ada1dbe
MQTT Broker: mqtts-up.iot.arduino.cc:8884
WiFi status ESP: 255
Voltage: 1.70
Current: 9.80
Power: 16.66
Energy: 0.01
Temp: 30.31
1.70
Saving:
0.00
Connected to "Redmi K40"

```

自动滚屏 Show timestamp 换行符 115200 波特率 清空输出

Figure 24: 显示已连接上 Wi-Fi

- 打开仪表板

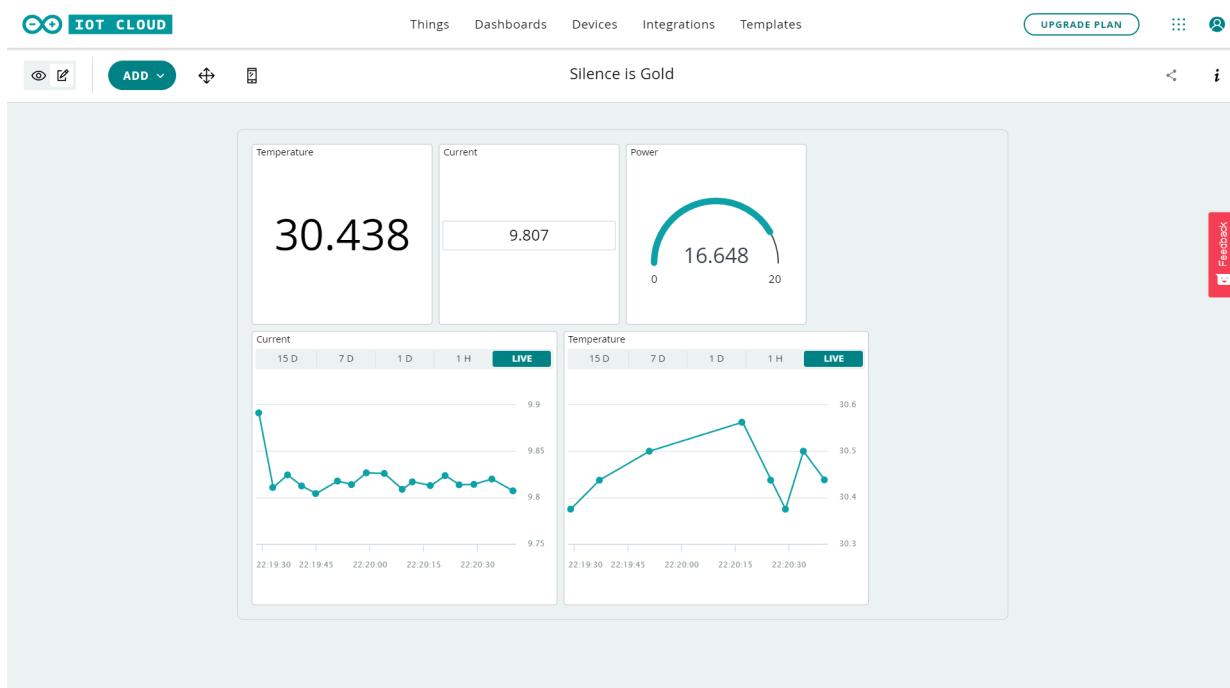


Figure 25: 已有数据显示

5.3 通过 Wi-Fi 或窄带物联网 (NB-IoT) 上传数据至私有云服务器

窄带物联网 (Narrow Band Internet of Things, NB-IoT) 成为万物互联网络的一个重要分支。NB-IoT 构建于蜂窝网络，只消耗大约 180kHz 的带宽，可直接部署于 GSM 网络、UMTS 网络或 LTE 网络，以降低部署成本、实现平滑升级。

NB-IoT 是 IoT 领域一个新兴的技术，支持低功耗设备在广域网的蜂窝数据连接，也被叫作低功耗广域网 (LPWAN)。NB-IoT 支持待机时间长、对网络连接要求较高设备的高效连接。据说 NB-IoT 设备电池寿命可以提高至少 10 年，同时还能提供非常全面的室内蜂窝数据连接覆盖。



Figure 26: NB-IoT 技术

5.3.1 应用场景

该设备可以通过两条途径将数据上传至云服务器，有适合家庭能耗检测的 WiFi 连接，还有适合野外无人值守的 NB-IoT 低功耗连接。



Figure 27: 野外太阳能发电站

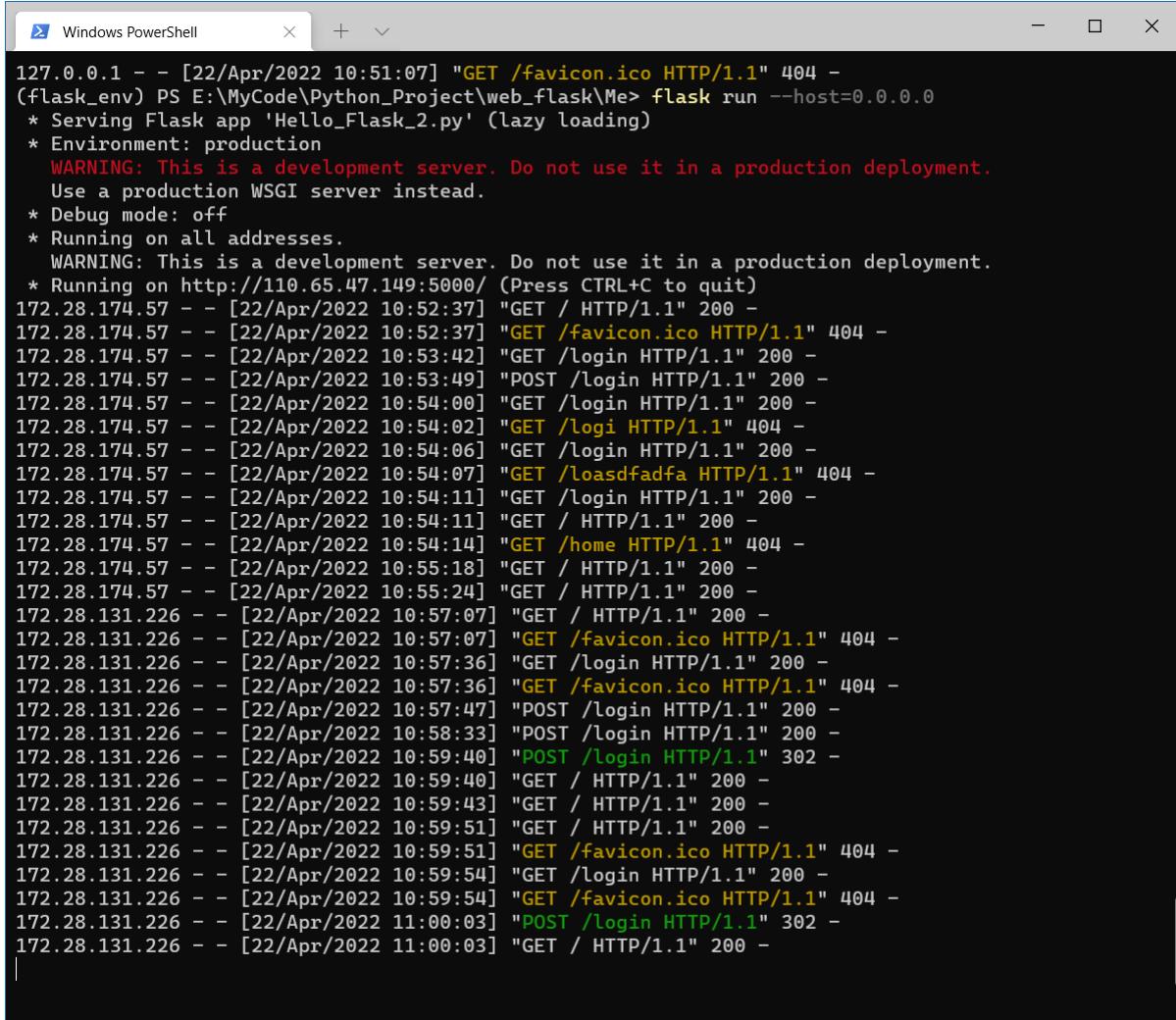
5.3.2 关键技术

窄带物联网（Narrow Band Internet of Things, NB-IoT）成为万物互联网络的一个重要分支。NB-IoT 构建于蜂窝网络，只消耗大约 180kHz 的带宽，可直接部署于 GSM 网络、UMTS 网络或 LTE 网络，以降低部署成本、实现平滑升级。

NB-IoT 是 IoT 领域一个新兴的技术，支持低功耗设备在广域网的蜂窝数据连接，也被叫作低功耗广域网（LPWAN）。NB-IoT 支持待机时间长、对网络连接要求较高设备的高效连接。据说 NB-IoT 设备电池寿命可以提高至少 10 年，同时还能提供非常全面的室内蜂窝数据连接覆盖。

5.3.3 WiFi 连接

使用 WiFi，开发板就像普通的上网设备一样接入互联网，能够很方便地使用 HTTP request 方法来向服务器传输数据。这里使用了 Flask 作为 Web application 还有内置的 Werkzeug 作为 Web 服务器网关接口。下面是 Windows 环境下 Web 服务的测试。



A screenshot of a Windows PowerShell window titled "Windows PowerShell". The window displays a log of HTTP requests from various IP addresses to a Flask application running on port 80. The log includes details like the request method (GET or POST), URL, and status code (e.g., 404, 200). Some lines in the log are highlighted in yellow and green, likely indicating specific requests or errors.

```

127.0.0.1 -- [22/Apr/2022 10:51:07] "GET /favicon.ico HTTP/1.1" 404 -
(flask_env) PS E:\MyCode\Python_Project\web_flask\Me> flask run --host=0.0.0.0
* Serving Flask app 'Hello_Flask_2.py' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://110.65.47.149:5000/ (Press CTRL+C to quit)
172.28.174.57 -- [22/Apr/2022 10:52:37] "GET / HTTP/1.1" 200 -
172.28.174.57 -- [22/Apr/2022 10:52:37] "GET /favicon.ico HTTP/1.1" 404 -
172.28.174.57 -- [22/Apr/2022 10:53:42] "GET /login HTTP/1.1" 200 -
172.28.174.57 -- [22/Apr/2022 10:53:49] "POST /login HTTP/1.1" 200 -
172.28.174.57 -- [22/Apr/2022 10:54:00] "GET /login HTTP/1.1" 200 -
172.28.174.57 -- [22/Apr/2022 10:54:02] "GET /logi HTTP/1.1" 404 -
172.28.174.57 -- [22/Apr/2022 10:54:06] "GET /login HTTP/1.1" 200 -
172.28.174.57 -- [22/Apr/2022 10:54:07] "GET /loasdafadfa HTTP/1.1" 404 -
172.28.174.57 -- [22/Apr/2022 10:54:11] "GET /login HTTP/1.1" 200 -
172.28.174.57 -- [22/Apr/2022 10:54:11] "GET / HTTP/1.1" 200 -
172.28.174.57 -- [22/Apr/2022 10:54:14] "GET /home HTTP/1.1" 404 -
172.28.174.57 -- [22/Apr/2022 10:55:18] "GET / HTTP/1.1" 200 -
172.28.174.57 -- [22/Apr/2022 10:55:24] "GET / HTTP/1.1" 200 -
172.28.131.226 -- [22/Apr/2022 10:57:07] "GET / HTTP/1.1" 200 -
172.28.131.226 -- [22/Apr/2022 10:57:07] "GET /favicon.ico HTTP/1.1" 404 -
172.28.131.226 -- [22/Apr/2022 10:57:36] "GET /login HTTP/1.1" 200 -
172.28.131.226 -- [22/Apr/2022 10:57:36] "GET /favicon.ico HTTP/1.1" 404 -
172.28.131.226 -- [22/Apr/2022 10:57:47] "POST /login HTTP/1.1" 200 -
172.28.131.226 -- [22/Apr/2022 10:58:33] "POST /login HTTP/1.1" 200 -
172.28.131.226 -- [22/Apr/2022 10:59:40] "POST /login HTTP/1.1" 302 -
172.28.131.226 -- [22/Apr/2022 10:59:40] "GET / HTTP/1.1" 200 -
172.28.131.226 -- [22/Apr/2022 10:59:43] "GET / HTTP/1.1" 200 -
172.28.131.226 -- [22/Apr/2022 10:59:51] "GET / HTTP/1.1" 200 -
172.28.131.226 -- [22/Apr/2022 10:59:51] "GET /favicon.ico HTTP/1.1" 404 -
172.28.131.226 -- [22/Apr/2022 10:59:54] "GET /login HTTP/1.1" 200 -
172.28.131.226 -- [22/Apr/2022 10:59:54] "GET /favicon.ico HTTP/1.1" 404 -
172.28.131.226 -- [22/Apr/2022 11:00:03] "POST /login HTTP/1.1" 302 -
172.28.131.226 -- [22/Apr/2022 11:00:03] "GET / HTTP/1.1" 200 -

```

Figure 28: Windows 环境下 Web 服务的测试

Flask 框架能够接受 GET 和 POST 请求，并解析，在开发板上使用 HTTP 请求，首先设定主机名称，可以替换为本地电脑的 ipv4 地址或者云服务器的 ipv4 地址

```

1 int    HTTP_PORT    = 80;
2 String HTTP_METHOD = "GET"; // or "POST"
3 char   HOST_NAME[] = "example.phpoc.com"; // hostname of web server
4 String PATH_NAME   = "";

```

检测连接情况并在串口向电脑打印信息

```

1 if( client.connect(HOST_NAME, HTTP_PORT) ) {
2     Serial.println( "Connected to server" );
3 } else {
4     Serial.println( "connection failed" );
5 }

```

把需要向服务器发送的信息写入字符串中，按照服务器能解析的格式

```
1 int temp = // read from sensor
2 int humi = // read from sensor
3 String queryString = String(" ?temperature=") + String(temp) + StPOS
```

通过 client 对象的 println 方法向服务器发送 POST 信息

```
1 // send HTTP header
2 client.println("POST" + PATH_NAME + "HTTP/1.1");
3 client.println("Host:" + String(HOST_NAME));
4 client.println("Connection: close");
5 client.println(); // end HTTP header
6
7 // send HTTP body
8 client.println(queryString);
```

最后检查服务器返回的值并向串口打印

```
1 while (client.available())
2 {
3     // read an incoming byte from the server and print them to serial monitor:
4     char c = client.read();
5     Serial.print(c);
6 }
7
8 if (!client.connected())
9 {
10    // if the server's disconnected, stop the client:
11    Serial.println("disconnected");
12    client.stop();
13 }
```

在服务端则需要解析 esp 32 发送的 POST 请求，读取响应的键值对，然后存储到相应的 SQL 服务器中。

以下是服务端的部分代码，具体任务就是接受来自 ESP 32 的 POST 请求后解析，读取出电压、电流、温度、ESP32 的时间，再加上服务端的时间，存放到已经建好的 SQL 数据库中。

```
1 app.route('/data', methods=['POST'])
2 def data():
3     error = None
4     if request.method == 'POST':
```

```

5     temperature = request.form['temperature']
6     current = request.form['current']
7     voltage = request.form['voltage']
8     esp_time = request.form['esp_time']

```

5.3.4 NB-IoT 连接

使用 NB-IoT 模块时，不再通过 HTTP 协议来传输数据，而是使用 MQTT 协议来通讯。MQTT（消息队列遥测传输协议），是一种基于发布/订阅（publish/subscribe）模式的“轻量级”通讯协议，该协议构建于 TCP/IP 协议上，可以以极少的代码和有限的带宽，为连接远程设备提供实时可靠的消息服务。正适合为对带宽和时延要求不高的物联网。

实现 MQTT 协议需要客户端和服务器端通讯完成，在通讯过程中，MQTT 协议中有三种身份：发布者（Publish）、代理（Broker）（服务器）、订阅者（Subscribe）。在这个项目中，ESP 32 就是发布者，MQTT 协议支持者是代理，接收数据的服务器是订阅者。

我们使用了移远 BC26 模块，主串口使用 AT 命令通信和数据传输。连接到 ESP 32 的 RXD 和 TXD 两个 pin 口上就能与 ESP 32 相互通信。

ESP 32 通过 Serial.println() 向 BC26 模块发送字符串数据，BC26 能接收数据然后向代理（Broker）发送数据，最后转发到服务端。

服务端的代码逻辑与 WiFi 连接的类似，也是接受数据然后存储到 SQL 数据库中。

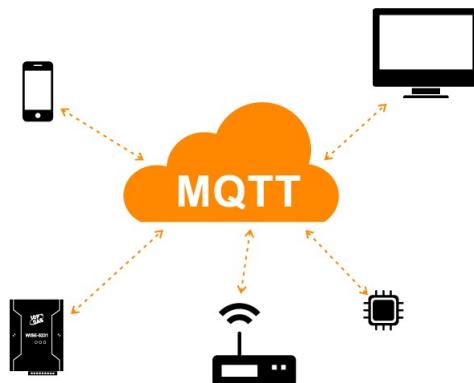


Figure 29: 消息队列遥测传输协议 MQTT

1. 实现 ESP32-S2 搭载 NB-IoT BC26 模块进行低功耗网络通信 ESP32 本身集成了蓝牙和 WiFi，为了使其接入 5G 模块，本组改装 ESP32 使其可与 NB-IoT 模块相连接。我们用 TXD、RXD、GND、3V3 引脚外接 BC26 无线通信模块。

在本系统中，ESP32 的 5G 信号通过 NB-IoT 技术与 5G 基站连接。

GPRS/EDGE 850/900/1800/1900Mhz 通过 AT 命令控制 ESP32 有 3 个 UART(UART0, UART1, and UART2)，但 ESP32-S2 仅有 2 个 UART (UART0 and UART1)，我们使用了 UART1。

具体步骤如下：

- (a) 定义使用的 UART 为 UART1

- (b) 判断使用的 UART_NUM_x 是否被占用
- (c) 设置 UART 参数，主要设置波特率，常用 115200, 9600
- (d) 设置通信引脚 PIN 设置 Tx、Rx、RTS 和 CTS 信号
- (e) 安装 UART 驱动程序
- (f) 发送和接收数据

```

1 char tx_data[512] = {0};
2 // assignment
3 uart_write_bytes(uart_num, (const char *) tx_data, strlen(tx_data));
4 // send data
5 int len = 0;
6 ESP_ERROR_CHECK(uart_get_buffered_data_len(uart_num, (size_t *)&len));
7 // Check whether the receiving buffer has returned data
8 char rx_data[512] = {0};
9 uart_read_bytes(uart_num, rx_data, len, 100);
10 // read data

```

至此，实现了 ESP32-S2 和 NB-IoT BC26 模块的连接。

2. 在平台创建相关产品和设备

首先登入平台-物联网平台-设备管理-产品，创建产品，再添加设备，设备添加好之后，点击 Device Secret 查看，可以看到三元素的相关参数。对于 Product Key 与 Device Secret 都是非常重要的参数，在设备登录的时候，是需要使用的。这个设备所建立的设备权限可以让此设备实现对数据的发布与订阅功能，根据实际需要进行填写相关的需求。

3. BC26 与云端连接

来设置让 BC26 通过 MQTT 协议登录到服务器，并实现数据的发布与订阅功能。查询到 IP 之后，表明已经可以正常注网了，那么就可以进行下一步的操作了。对于操作 BC26 的 MQTT 可以参考 BC26 的 MQTT 官方的使用说明书对相关的指令进行了解。可以分别看到，上面的 MQTT 参数配置是和设备的参数是需要对应的，分别是对应产品 key，设备名称以及设备密钥。一一对应，类似于账号密码以及登录 ID 等。登录私人云服务器的地址，MQTT 的端口一般是 1883，不过 BC26 也支持域名，用户采用域名的方式登录也是可以的。

4. 实现数据订阅与发布

那么就可以进行下面的数据订阅与发布，发布数据用户需要用 A-LINK 格式要求如下：

```

1 {"id": "26", "version": "1.0", "params":{ "CurrentTemperature":{ "value":23}, "RelativeHumidity
2 ":{ "value":58} }, "method": "thing.event.property.post" }
3 // Serial port sends data to module:

```

```

4 AT+QMTPUB=0,0,0,0, "/sys/a1qmGxDM8cd/mzhtest001/thing/event/property/
   post", "
5 { "id": "26", "version": "1.0", "params":{ "Voltage":{ ``value":23}, "Current
   ":{ "value":58} }, "method": "thing.event.property.post" }"

```

5. 设备属性上报

正常上报之后，会显示发布成功，返回 +QMTPUB:0,0,0 代表上报成功，可以看到设备已经显示在线。数据格式选择 ALINK 协议。设备上报的时候遵循 ALINK 协议，云平台就可以解析数据。私人云服务器的 ALINK 是基于 MQTT 协议的，我们用 MQTT.fx 客户端软件测试数据。这样可以排除硬件的故障干扰，先把数据上云的流程跑通。根据设备 Thermometer_01 的三元组信息生成基本信息。然后，打开 MQTT.fx 填入相关字段。根据文档中的实例和我们自主定义的数据格式，组成一个 JSON 格式的数据使用 AT 指令控制 BC26 模块上报属性。为了模块的稳定运行，上电后 60 秒左右送 AT 指令。至此，BC26 接入平台成功实现。

5.3.5 测试结果

输入帐号密码登录配置好的私人云服务器 “SolarPanelManagement”

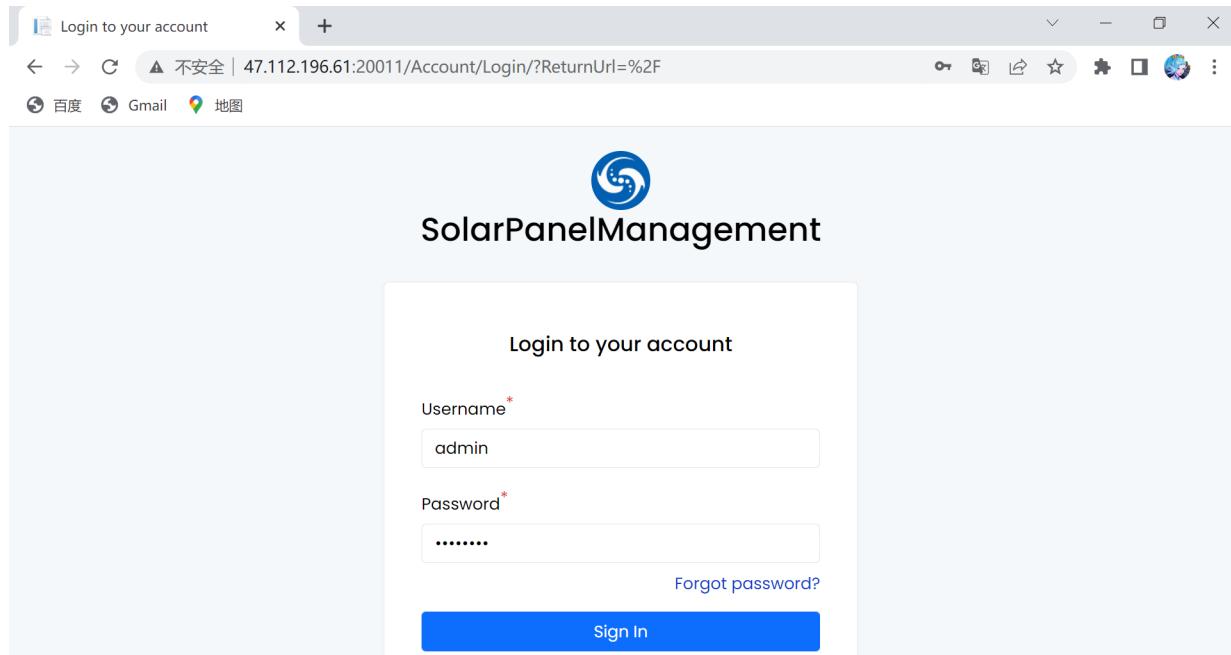


Figure 30: 云端登录界面

可以看到监测系统前端界面如图所示：

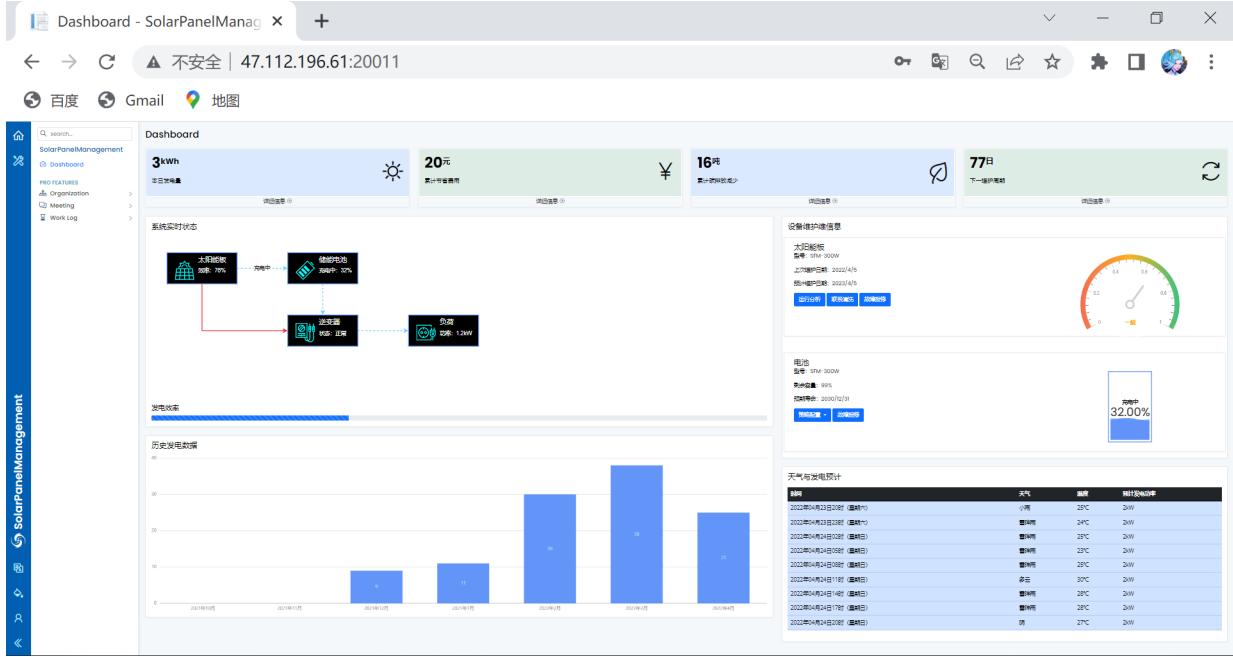


Figure 31: 监测系统

用户可以实时查看太阳能板工作状况，累计节省电量，累计碳排放减少等信息。还能根据天气情况预测发电量，并与实际值对比排查故障。

测试成功。

6 数据处理与分析

我们可以基于当地供电局电价及中华人民共和国生态环境部印发的火力发电量和碳排放量的换算标准，我们可以在 Arduino IDE 通过编辑代码显示一些除电学参量以外的数据，例如发电量、累计节省费用、累计碳排放减少等，来使客户基于数据直观评判太阳能板的经济效益、环保效益，做到促进太阳能板的安装及科学使用。



Figure 32: DashBoard 上部分个性化数据

我们还可以导出云端观测的太阳能板实时功率数据进行分析，呈现不同时段的功率值。

20W太阳能板实时功率监测

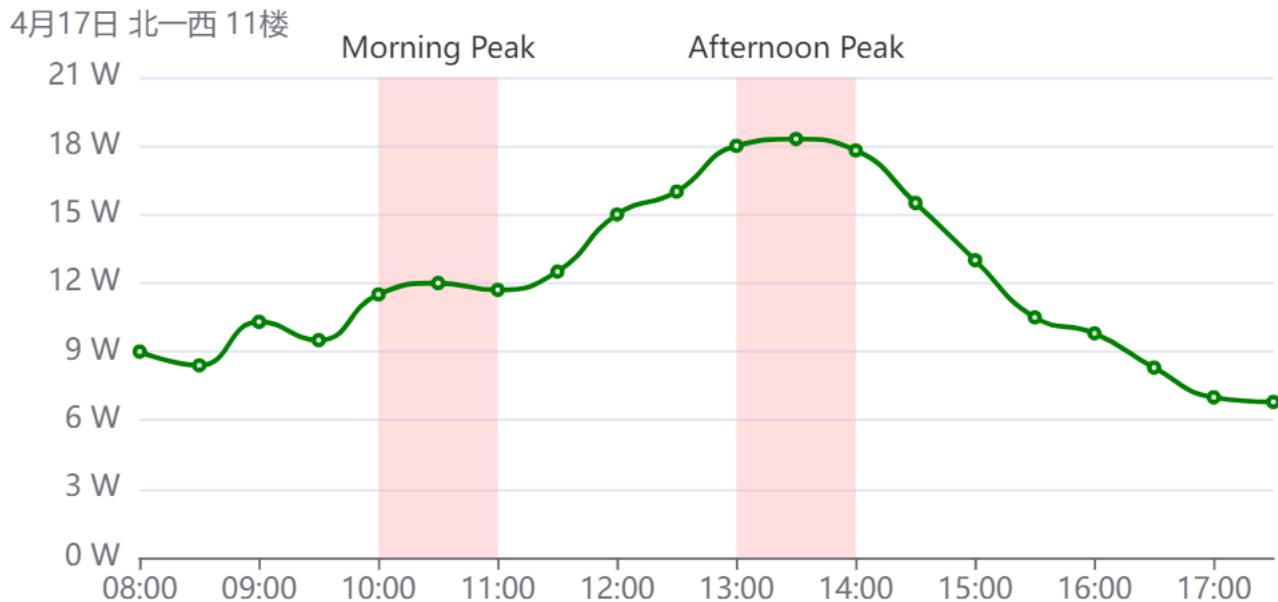


Figure 33: 监测折线图

除上述数据采集及分析以外，我们还可以充分利用私人云服务器灵活可调的特点，根据自己/客户的需求设置前端界面，实现基于观测到的电学参数和一些预测算法对太阳能板的寿命进行预测等功能。

7 特色与创新

在这个万物互联飞速发展的社会，伴随着经济全球化进程，电子信息技术在人们的生活中发挥着越来越重要的作用。NB-IoT 网络和 5G 的融合使其具有更多复杂的特性，也使其为人类的发展创造出更多的社会价值和经济价值。基于 NB-IoT 网络和 5G 技术，我们的作品具有如下特色：

(1) 通过 PCB 板的设计，整个监测装置几近手机大小，只需要外加金属外壳，连接输出输入两端的四个接口，即可完成监测并上网，实现了用户端的“即插即用，零配置”。

(2) 运营成本低。基于 NB-IoT 网络具备的低功耗、广覆盖、低成本、大容量等优势，我们设计的监测系统不仅成本低廉，还更加节能环保。此外，5G 物联网技术通过实现实时远程状态监控来解决预测性维护的挑战。公司可以使用物联网健康监控进行直接维护，而不是每月或每个季度维护一次硬件。通过物联网设备现在可以实时监控关键性能指标，并在问题开始时发送警报，减少了运营成本。

(3) 不受地域限制。因为离网光伏发电装置的应用场景经常在山区、海上、高原、田野等网络没有覆盖或覆盖不全面的地方，相比 WIFI、蓝牙、ZigBee 等短距离通信技术，NB-IoT 网络具备广覆盖、可移动以及大连接数等特性，能够带来更加丰富的应用场景。

(4) 良好的产业化前景。如今新能源产业已经进入到快速、良性的发展道路，产业模式的转变加快

了行业信息化进程，我们的作品就是该行业信息化的体现，虽然目前的成品比较简单，但我们认为它还有很大的发展空间，具有良好的产业化前景。

(5) Web 可个性化修改前端内容，与客户需求实时对接，具有数据安全性高，IT 基础架构可控制能力强、合规等特点，更能满足用户的特定需求。

8 问题与展望

5G 时代之下，我们正在见证物联网的发展，也实实在在的享受着物联网带给我们的便利，我们也相信物联网是面向未来、改变未来的工程。基于物联网的离网光伏发电装置监测系统的设计和实现涉及多方面的理论、方法和技术，本系统还有许多问题需要解决，需要在实际应用中不断积累和完善，在以下几个方面，还可以做进一步的研究和开发：

(1) 由于作品完成时间有限，又受到疫情影响不宜出校，我们的作品目前进行测试主要是依靠购买的小型太阳能电池板，还没有在实际生产生活中用的光伏发电组件上进行测试。对此我们希望可以通过更多的实践和检验来完善作品。

(2) 因为我们对物联网相关理论知识等了解较少，所以我们想要加强这方面知识的学习，再深入研究光伏发电的云集成技术，实现光机电设备的群控、联调，既关注小型离网光伏发电系统的监测情况，又让工具更大范围扩展到行业尤其是生产生活领域。

(3) “双碳”目标催生新能源行业发展新机遇，新能源行业环境继续向好，行业发展前景可期。为了响应国家的号召，也为了在节能减排的时代号召中尽一份自己的力量，我们认为可以扩大作品的使用范围，可以考虑加入蓄电池储存电能，加逆变器变直流为交流保护负载，完善本系统使其兼备离网和并网光伏发电装置的监测能力，为用户带来更多便捷。

9 参考文献

- [1] S.N Hiroki Harada, K.Aoyagi and H.Takahashi “5G Evolution Directions and Standardization Trends,” Special articles on 3Gpp release 16 Standardization Activities,vol 22, No 3 Jan 2021.
- [2] Caso Giuseppe, Kousias Konstantinos, Alay, Brunstrom Anna, Neri Marco ”NB-IoT Random Access: Data-driven Analysis and ML-based Enhancements” IEEE Internet of Things Journal (2021)
- [3] S.A. Gbadamosi, G.P. Hancke, A.M. Abu-Mahfouz “Building Upon NB-IoT Networks: A Roadmap Towards 5G New Radio Networks” .IEEE Access, 8 (2020), pp. 188641-188672
- [4] 李琴, 潘三博. 基于 NB - IoT 的光伏电站监测系统. 仪表技术与传感器.2021, 第八期:59-62
- [5] 党凯强, 姚金杰, 贺冠华, 郭华, 张俊虎. 一种低功耗 NB-IoT 远程监测终端设计. 国外电子测量技术.2020,39(12):136-140
- [6] 王能辉, 胡国强. 基于 NB-IoT 的农田远程监测系统的设计. 陕西农业科学. 2017,63(12):82 - 85