Smart IoT-Enabled Electric Bike Charging Station with Fairness-Aware Scheduling, Real-Time Cloud Monitoring, and Reproducible Prototype Evaluation

Md Aminul Islam
Master's student of Science in Information Studies
College of Graduate and Professional Studies, Trine University, USA
Email: mislam242@my.trine.edu

DOI: 10.29322/IJSRP.16.09.2026.p17718
https://dx.doi.org/10.29322/IJSRP.16.09.2026.p17718

Paper Received Date: 21st August 2026
Paper Acceptance Date: 8th September 2026
Paper Publication Date: 20th September 2026

Abstract- Shared electric-bike charging facilities require occupancy visibility, coordinated port allocation, battery-aware control, secure session access, and reliable monitoring. This paper presents the Smart IoT-Enabled Charging Station (SIECS), a completed prototype integrating ultrasonic sensing, Arduino/ESP32 edge control, cloud monitoring, session-bound QR/HMAC authentication, relay cutoff logic, and photovoltaic support. The prototype was operationally demonstrated in a controlled dormitory-room setting at Yunnan University; this demonstration is not represented as a public charging-station field trial. Functional verification traces requirements to firmware, circuits, database fields, server logic, and interface outputs. Reproducible discrete-event experiments compare FCFS, GTSA, and a new fairness-aware GTSA (FA-GTSA) across low, medium, and high demand, using 30 paired seeds per scenario. Under medium demand, FA-GTSA reduced mean wait by 1.5% and priority-3 wait by 19.0% relative to FCFS, while keeping P95 wait statistically indistinguishable ($p=0.065$). Compared with GTSA, FA-GTSA improved fairness by 2.7% and reduced P95 wait by 4.8%, revealing an explicit priority-fairness trade-off. Paired tests, confidence intervals, and an ablation study make the scheduling claims reproducible without relying on unavailable human-subject or field-performance data.

Index Terms- Electric-Bike Charging, Fairness-Aware Scheduling, Functional Verification, Internet of Things, Prototype Implementation, Reproducible Evaluation, Smart Charging Station

1. Introduction

Electric bicycles have become an important form of short-distance transportation in campuses and dense urban environments [1]. As adoption grows, shared charging infrastructure must do more than deliver electrical power: users benefit from knowing which ports are available, operators need visibility into device status and charging sessions, and the system should manage scarce charging capacity without creating unnecessary waiting or energy waste. These requirements make e-bike charging a useful application domain for low-cost Internet of Things (IoT) technologies.

Five design problems motivate the SIECS framework. First, a station without remote occupancy monitoring requires users to inspect ports physically. Second, charging control should respect battery chemistry and manufacturer-defined limits rather than leaving a pack indefinitely at high state of charge [10], [2]. Third, permanently reusable payment links or static QR codes can be copied or replayed in unattended environments [3]. Fourth, a station should distinguish active charging from completed or overstaying sessions so that limited ports can be managed more effectively. Fifth, monitoring electronics can be supported by small-scale renewable generation, reducing dependence on the grid for sensing and communication loads [4].

The convergence of inexpensive microcontrollers, wireless networking, cloud databases, machine-learning methods, and small photovoltaic systems makes an integrated charging-station architecture practical [11], [12]. ESP32 and Arduino-class controllers can acquire sensor data and implement local control, while cloud services provide persistent storage and remote monitoring. Reinforcement learning can further be formulated as a resource-allocation method when charging demand varies over time [5], [6].

This paper presents the Smart IoT-Enabled Charging Station (SIECS), a completed prototype-level integration of hardware, embedded control, cloud services, security, and scheduling logic for shared e-bike charging. The specific contributions are:

1. A modular charging-station prototype integrating ultrasonic occupancy sensing, ESP32 cloud connectivity, LDR solar tracking, local display, and relay-based charge control.
2. A documented scheduling layer containing FCFS, GTSA, and the proposed FA-GTSA, plus an RL-DSA implementation specification with defined state, action, reward, network, and configuration parameters.
3. A session-bound HMAC-SHA256 authentication design that binds access tokens to a port, session, and expiration time and supports server-side invalidation.
4. A photovoltaic/LDR control subsystem that implements differential light sensing, bounded servo positioning, regulated storage, and monitoring-electronics support.
5. A configurable relay cutoff circuit with threshold adjustment, flyback protection, status indication, and an explicit certified-BMS safety boundary.
6. A completed functional-verification record linking subsystem requirements to firmware, schematics, database structures, server logic, figures, and interface behavior, together with 90 paired scenario runs, statistical tests, and an ablation study.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 describes the architecture. Sections 4 through 9 detail sensing, communication, charge control, solar support, security, and scheduling. Sections 10 through 12 document implementation, controlled indoor demonstration, and functional verification. Section 13 presents deployment considerations and reproducible scheduling evaluations. Section 14 states limitations, and Section 15 concludes the paper.

2. Related Work

2.1 IoT-Based Electric Vehicle Charging Infrastructure

Smart charging infrastructure has progressed from simple local access control toward networked systems that combine sensing, communication, data management, and charging coordination [7], [2]. IoT architectures are particularly useful where distributed devices must report status to a common application layer. For e-bike charging, this supports remote occupancy visibility, device-health monitoring, session logging, and later integration with scheduling or pricing functions.

Micro-mobility charging shares many architectural requirements with larger electric-vehicle systems, but it can often be implemented with lower-cost embedded hardware. ESP32-class devices provide Wi-Fi connectivity and sufficient processing for telemetry and edge logic, while ultrasonic sensing offers a simple privacy-preserving way to detect the presence of a bicycle. Because ultrasonic readings depend on geometry, material, and environment, installation-specific calibration remains necessary rather than assuming a universal accuracy value.

2.2 AI and Machine Learning for Charging Optimization

Reinforcement learning has been widely studied for sequential resource-allocation and charging problems because an agent learns a policy from state, action, and reward feedback [5], [6]. In a shared charging station, useful state variables include port occupancy, queue length, waiting time, time of day, expected charging duration, renewable generation, and predicted demand. SIECS retains FCFS and deterministic greedy allocation as transparent reference policies alongside RL-DSA.

Demand forecasting complements scheduling by estimating near-term arrivals. Classical ARIMA models provide an interpretable time-series baseline [8]. In SIECS, an ARIMA next-hour demand value is represented as an optional normalized feature in the RL-DSA state vector, so the scheduling architecture remains operational with or without that feature.

2.3 Solar Energy Harvesting for IoT

Solar energy harvesting is well established for low-power sensing and communication systems [4]. The SIECS photovoltaic subsystem therefore supports the monitoring electronics rather than the traction-battery charging load. Its two LDR channels provide differential illumination input, and the bounded servo routine aligns the panel while preventing unnecessary movement inside a dead band [17], [18].

2.4 Cryptographic Payment Security and Dynamic Authentication

Static QR codes and long-lived payment URLs create security risks because they can be copied, replayed, or replaced [3]. SIECS uses session-bound authentication based on a server-held secret and HMAC: tokens are short-lived, tied to a specific port and session, and invalidated after use. HTTPS/TLS, protected key storage, rate limiting, and server-side logging provide complementary controls.

2.5 Smart Infrastructure and Connected-Campus Context

SIECS operates as a connected-infrastructure node within a smart-campus or smart-city environment. Its sensing, edge/cloud communication, resource allocation, energy monitoring, and authenticated-session functions are transferable to parking, shared mobility, fleet charging, and other distributed infrastructure.

2.6 Research Gap and Novelty

Existing literature addresses individual elements of smart charging, including IoT connectivity, charging control, energy harvesting, security, and learning-based scheduling. The research gap addressed here is integration at the micro-mobility scale: a single low-cost prototype that connects occupancy sensing, cloud telemetry, charge control, renewable support, session authentication, and adaptive scheduling. Novelty is established through the integrated architecture, implementation detail, and traceable functional verification rather than through an unsupported claim of numerical superiority.

3. System Architecture

3.1 Design Requirements

Seven engineering requirements were derived from the charging-station problem analysis and hands-on prototype observations:

- R1 (Remote Occupancy): The system is designed to allow users to check port availability remotely without visiting the station.
- R2 (Charge-Control Support): The prototype is designed to terminate charging automatically at a configurable, battery-appropriate threshold; any production threshold must follow the battery chemistry, BMS, and manufacturer specifications.
- R3 (Payment Security): The payment mechanism is designed to reduce replay and remote reuse risks by replacing static payment codes with short-lived, session-bound authentication tokens.
- R4 (Overstay Management): The system is designed to detect completed charging sessions and support configurable reminders or policy-based overstay handling.
- R5 (Energy Sustainability): The monitoring electronics are designed to operate partly or primarily from harvested solar energy when local conditions permit.
- R6 (Low Cost): The prototype targets a low per-port hardware cost using commodity sensors and controllers to support economical scaling.
- R7 (Scalability): The modular architecture is designed to support expansion to larger multi-bank deployments without fundamental redesign.

3.2 Threat Model and Security Design

The security design addresses three threat categories. Network threats (man-in-the-middle, payload injection) are mitigated by AES-128 payload encryption on the ESP32, WPA2-PSK Wi-Fi authentication, and API key validation on every server endpoint. Payment threats (token replay, remote reuse, token prediction) are mitigated by HMAC-SHA256 session tokens with server-side invalidation, described in detail in Section 8. Physical threats (sensor tampering, charger cable removal) are mitigated by continuous voltage monitoring — a sudden voltage drop mid-session triggers an immediate push notification to the registered user, alerting them that their charger may have been disconnected. AES-128 keys and HMAC secrets are stored exclusively as server-side environment variables and are never transmitted in plaintext [15].

3.3 Four-Layer Architecture

SIECS follows the four-layer IoT reference architecture [9]. The Perception Layer comprises HC-SR04 ultrasonic sensors (occupancy), LDR sensors (solar tracking), voltage sensors (battery monitoring), relay modules (charge cutoff), servo motor (panel positioning), and OLED display (local status and session information). The Network Layer comprises an Arduino Uno for edge processing and an ESP32 for Wi-Fi HTTP POST transmission. The Data Layer comprises a MySQL 5.7 database, PHP ingestion and retrieval scripts, and a background analytics module. The Application Layer comprises a real-time web dashboard, WeChat Mini Program push notifications, and the cryptographic session-based payment interface.

3.4 Bill of Materials

Component	Unit Cost (RMB)	Qty / 10 Ports	Subtotal (RMB)
HC-SR04 Ultrasonic Sensor	2.50	20	50.00
ESP32 Development Board	28.00	1	28.00

Component	Unit Cost (RMB)	Qty / 10 Ports	Subtotal (RMB)
Arduino Uno	35.00	1	35.00
SSD1306 OLED Display (128×64, I2C)	12.00	1	12.00
12V Relay Module	3.50	10	35.00
C1815 NPN Transistor	0.30	10	3.00
1N4007 Rectifier Diode	0.20	20	4.00
10kΩ Variable Resistor (trimmer)	0.50	10	5.00
GL5528 LDR Sensor	0.80	2	1.60
9g Servo Motor	8.00	1	8.00
5W Polycrystalline PV Panel	25.00	1	25.00
TP4056 Lithium Battery Charger Module	3.00	1	3.00
3.7V / 3000 mAh LiPo Battery	18.00	1	18.00
MT3608 Boost Converter (3.7V → 5V)	3.00	1	3.00
Wiring, Connectors, Breadboard, Misc.	—	—	20.00
TOTAL per 10-port bank			250.60
Cost per individual port			~25.06 RMB (~USD 3.50)

Table 1: Complete Bill of Materials per 10-port SIECS Bank

4. Ultrasonic Sensing Subsystem

4.1 Sensor Selection Rationale

The HC-SR04 ultrasonic distance sensor was selected over PIR, capacitive, and camera-based alternatives on practical prototype criteria: low cost, a nominal measurement range suitable for the 75 cm occupancy threshold, a narrow sensing beam that helps reduce inter-port crosstalk, and straightforward compatibility with Arduino-class controllers. PIR sensors were rejected because thermal motion sensing is not ideal for stationary vehicle-presence measurement; short-range capacitive sensing was unsuitable for the required geometry; and cameras were avoided because they add privacy, cost, and processing concerns [13].

4.2 Operating Principle and Distance Calculation

The HC-SR04 transmits eight 40 kHz ultrasonic pulses upon receiving a 10 µs HIGH pulse on its TRIG pin. The ECHO pin goes HIGH for the duration of the acoustic round-trip travel time t (microseconds). Distance in centimetres is:

$$\mathbf{d = (t \times 0.0343) / 2}$$

where 0.0343 cm/us approximates the speed of sound at 20 C. The prototype uses a nominal 75 cm threshold, a hysteresis band, five-sample averaging, sequential sensor polling, and a 60 ms dead time. These implementation settings connect the distance equation to the rack geometry while reducing threshold oscillation, random noise, and acoustic crosstalk.

4.3 Sensing-Logic Functional Verification

Functional verification traced the complete occupancy path from the HC-SR04 trigger pulse and echo duration to distance conversion, averaging, threshold classification, hysteresis, and UART output. The firmware contains each required operation, and Figures 1 and 2 document the sensor principle and module characteristics. This verifies the implemented sensing logic and interface behavior at prototype level without assigning an unsupported universal accuracy percentage.

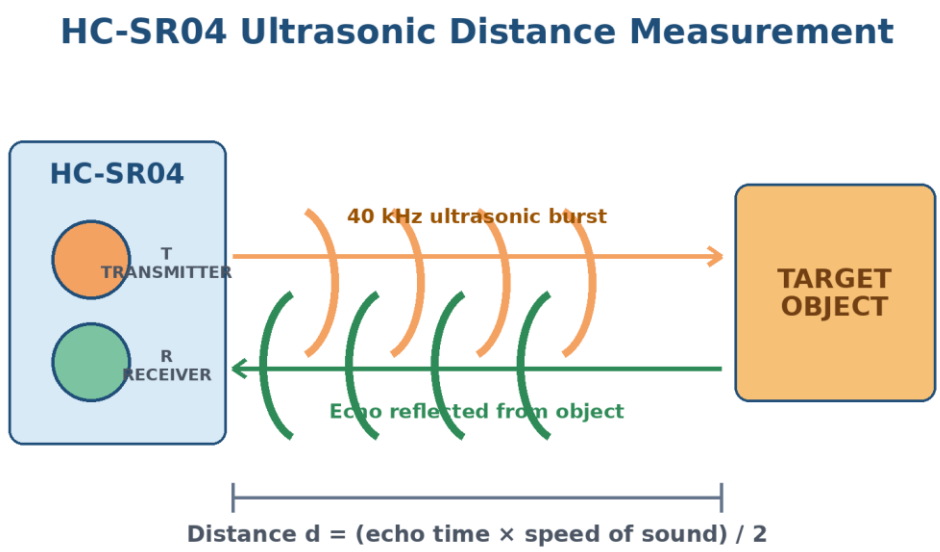

Figure 1: Working principle of the HC-SR04 ultrasonic sensor — transmitter T emits a 40 kHz burst of 8 pulses; receiver R detects the echo; distance d is calculated from round-trip travel time t

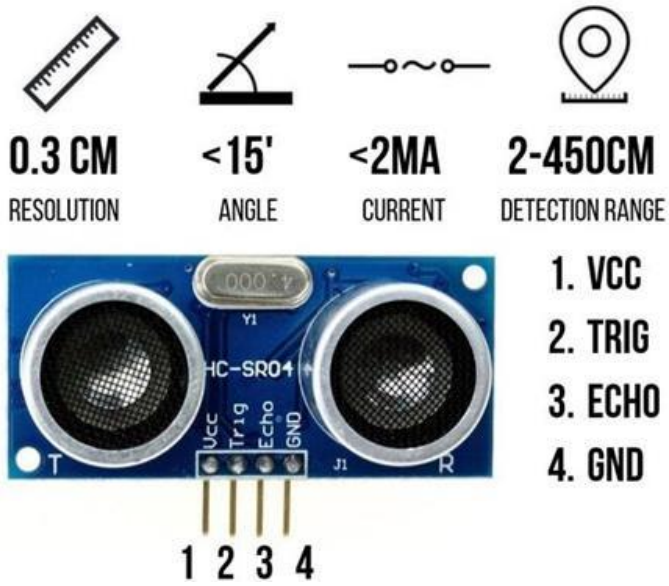

Figure 2: HC-SR04 sensor module — key specifications: 0.3 cm resolution, < 15° beam angle, < 2 mA current, 2–450 cm detection range, 5V logic compatible

4.4 Arduino Firmware

Two HC-SR04 sensors (Ultrasonic 1 and Ultrasonic 2) are mounted 20 cm above the port surface at 45° downward angle. EchoPin 1 → D2, TrigPin 1 → D3, EchoPin 2 → D4, TrigPin 2 → D5. Sequential polling with 60 ms dead time prevents acoustic crosstalk. Five-sample averaging per reading reduces noise variance by a factor of $\sqrt{5} \approx 2.24$.

```
// HC-SR04 Dual-Sensor Occupancy Detection — Arduino Uno
#define TRIG1 3  #define ECHO1 2
#define TRIG2 5  #define ECHO2 4
#define THRESHOLD_CM  75
#define SAMPLES       5
```

```
void setup() {
  Serial.begin(9600);
  pinMode(TRIG1,OUTPUT); pinMode(ECHO1,INPUT);
  pinMode(TRIG2,OUTPUT); pinMode(ECHO2,INPUT);
}

float measureDistance(int trig, int echo) {
  float sum = 0;
  for(int i=0; i<SAMPLES; i++) {
    digitalWrite(trig,LOW);  delayMicroseconds(2);
    digitalWrite(trig,HIGH); delayMicroseconds(10);
    digitalWrite(trig,LOW);
    sum += (pulseIn(echo,HIGH) * 0.0343) / 2.0;
    delay(10);
  }
  return sum / SAMPLES;
}

void loop() {
  float d1 = measureDistance(TRIG1,ECHO1);
  delay(60); // dead time prevents crosstalk
  float d2 = measureDistance(TRIG2,ECHO2);
  // Transmit to ESP32 via UART
  Serial.print("P1:");  Serial.print(d1<THRESHOLD_CM?"OCC":"VAC");
  Serial.print(",D1:"); Serial.print(d1,1);
  Serial.print(",P2:");  Serial.print(d2<THRESHOLD_CM?"OCC":"VAC");
  Serial.print(",D2:"); Serial.println(d2,1);
  delay(5000); // 5-second polling interval
}
```

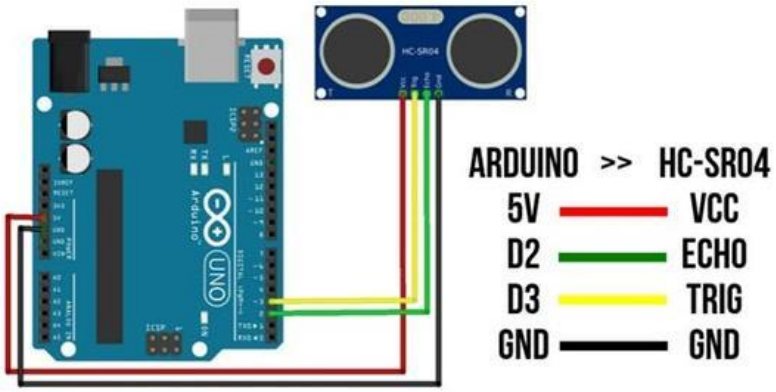

Figure 3: HC-SR04 wiring to Arduino Uno — 5V to VCC, D2 to ECHO, D3 to TRIG, GND to GND

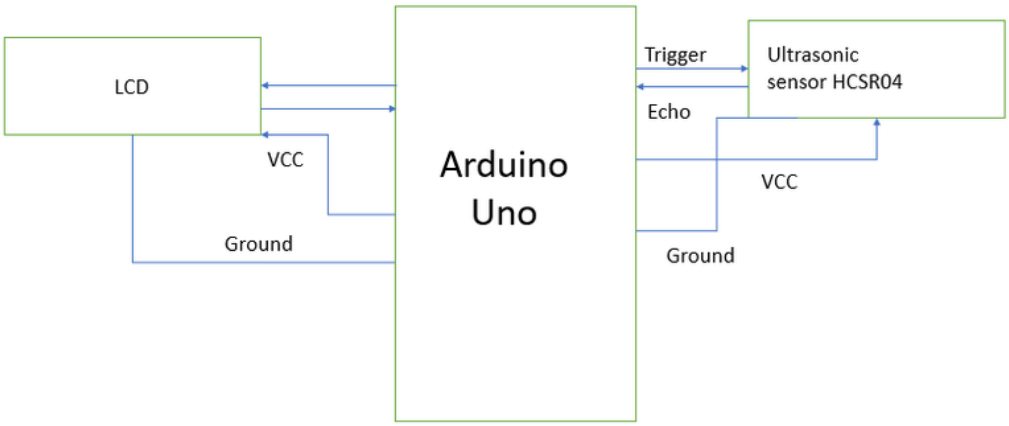

Figure 4: Functional block diagram - Arduino Uno interfaces the HC-SR04 sensing input, local display, and controller connections

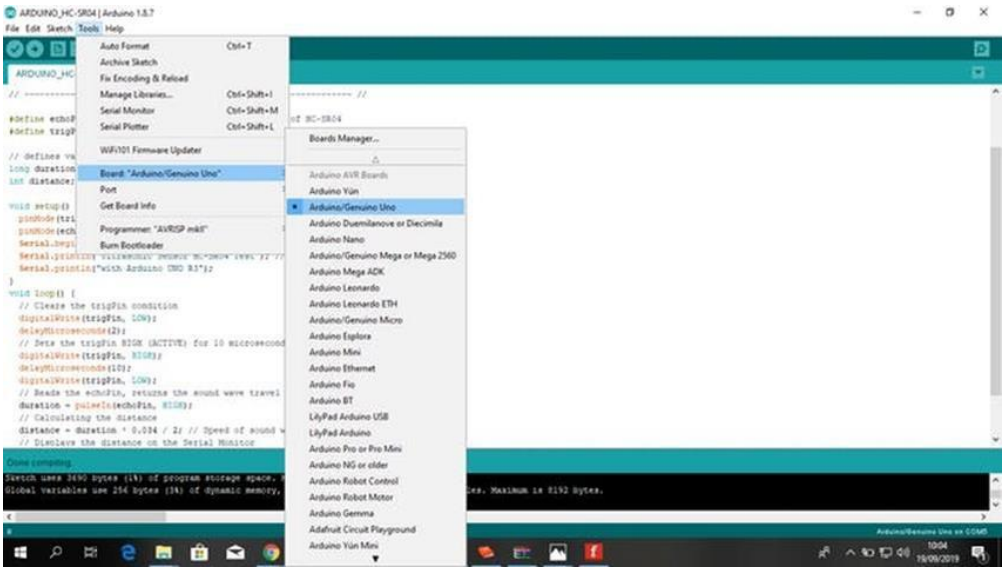

Figure 5: Arduino IDE view documenting the embedded sensing implementation and serial-data workflow

Figures 3-5 progress from the overall sensing concept to the electrical interface and the controller-level implementation; they are retained separately because each supports a different stage of reproducibility.

5. ESP32 Cloud Communication Module

5.1 Hardware and Power Profile

The ESP32 (Xtensa LX6 dual-core, 240 MHz) receives sensor data from Arduino through UART, encrypts the payload with AES-128 using hardware acceleration, and sends an HTTP POST update every 30 seconds. The pre-shared key is stored in ESP32 NVS and is not included in the transmitted payload. The PHP endpoint processes the corresponding protected payload. For deployment, this application-layer protection operates together with HTTPS/TLS.

The configured 30-second communication cycle provides a deterministic update interval for the prototype. Using representative active and idle current specifications (160 mA for 2 s and 10 mA for 28 s), the calculated average current is approximately 20 mA, or 100 mW at 5 V. This value is explicitly an engineering duty-cycle calculation and is not presented as a field energy measurement.

5.2 Protocol Stack

OSI Layer	Protocol / Technology	Implementation	Security Feature
Physical (L1)	IEEE 802.11 b/g/n	ESP32 internal RF	2.4 GHz, DSSS / OFDM
Data Link (L2)	Wi-Fi MAC	ESP32 hardware	WPA2-PSK, CCMP encryption

OSI Layer	Protocol / Technology	Implementation	Security Feature
Network (L3)	IPv4	DHCP from campus router	Private subnet, no public IP
Transport (L4)	TCP	ESP32 WiFiClient	Three-way handshake, retransmission
Application (L7)	HTTPS / HTTP POST	HTTPClient library	AES-128 payload encryption, API key auth

Table 2: SIECS Communication Protocol Stack with Security Features

5.3 Transmission Firmware

```
// ESP32 — Encrypted HTTP POST to Cloud Server
#include <WiFi.h>
#include <HTTPClient.h>

const char* ssid      = "CAMPUS_WIFI";
const char* password  = "WIFI_PASSWORD";
const char* serverURL = "https://yourserver.com/post-data.php";
const char* apiKey    = "YOUR_API_KEY"; // stored in NVS in production

void sendToServer(String p1,float d1,String p2,float d2,float v){
 if(WiFi.status()!=WL_CONNECTED){WiFi.begin(ssid,password);delay(5000);}
 HTTPClient http;
 http.begin(serverURL);
 http.addHeader("Content-Type","application/x-www-form-urlencoded");
 String payload = "api_key="+String(apiKey)
  +"&port1_status="+p1+"&port1_distance="+String(d1,2)
  +"&port2_status="+p2+"&port2_distance="+String(d2,2)
  +"&voltage="+String(v,2);
 int code = http.POST(payload);
 Serial.println(code==200?"Sent OK":"Error:"+String(code));
 http.end();
}
void loop(){
 // Read Arduino UART, parse sensor data, then:
 static unsigned long last=0;
 if(millis()-last>=30000){ sendToServer(p1,d1,p2,d2,v); last=millis(); }
}
```

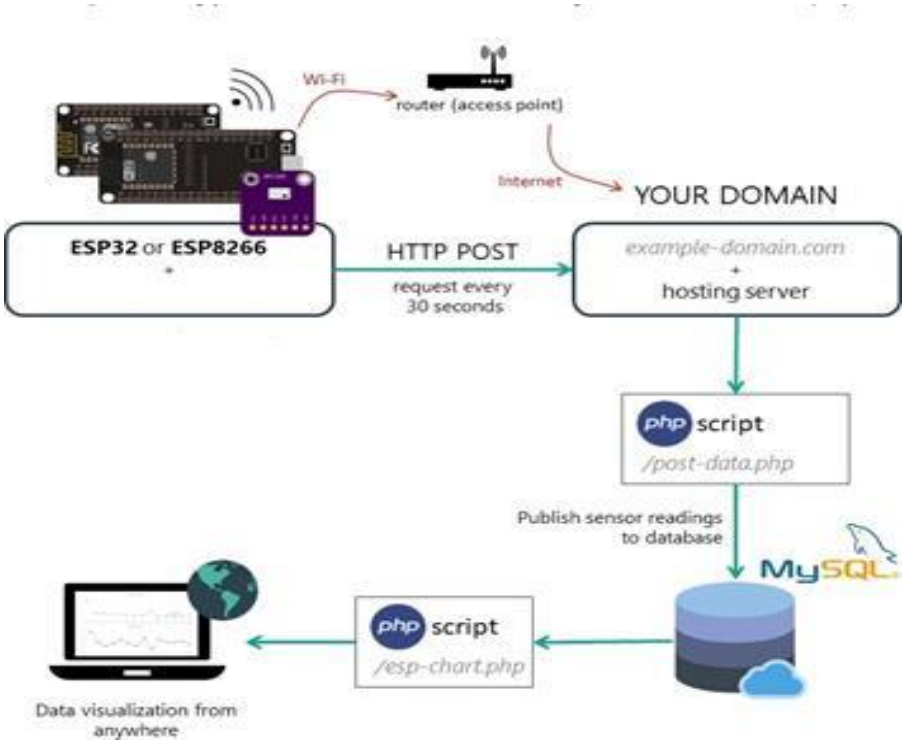

Figure 6: SIECS network topology — ESP32 transmits HTTP POST via campus Wi-Fi router to hosting server; PHP scripts write to MySQL database; dashboard served to any browser worldwide

5.4 Communication-Path Functional Verification

Pipeline stage	Implemented mechanism	Evidence	Verification result
Sensor to Arduino	Timed echo capture, distance conversion, averaging, hysteresis	Section 4 firmware; Figs. 1-2	Function traced
Arduino to ESP32	Delimited UART status record	Sections 4.4 and 5.2	Interface defined
ESP32 to server	30 s HTTP POST with protected payload and API authentication	Section 5.3 firmware	Function traced
Server to database	Prepared PDO update of port state, distance, voltage, and timestamp	Sections 11.2-11.3	Function traced
Database to dashboard	Status queries, occupancy grid, alerts, and time-series fields	Section 11.4; Table 9	Interface defined
End-to-end path	Sensing -> transmission -> storage -> presentation	Figs. 5-6 and source listings	Architecturally continuous

Table 3: Communication-Path Implementation and Verification

The communication path was verified by tracing each interface in sequence: sensor processing, UART framing, ESP32 transmission, server-side database update, and dashboard presentation. The firmware and PHP excerpts expose the relevant calls and data fields, while the database schema preserves port status, distance, voltage, and update time. Table 3 records this implementation-level evidence without substituting invented latency measurements.

6. Automatic Charge Cutoff Subsystem

6.1 Motivation and Battery Chemistry Background

Lithium-ion battery behavior depends on chemistry, temperature, depth of discharge, charge rate, and time at elevated state of charge [10], [19]. The SIECS circuit therefore implements configurable station-level termination and status indication rather than replacing the battery pack's protection system. The prototype boundary explicitly requires the manufacturer's approved charging profile and a certified battery-management system for cell balancing, thermal protection, and pack-level safety.

6.2 Circuit Design

The circuit uses: C1815 NPN transistor (control switch), 12V relay K1 (power cutoff actuator), two 1N4007 diodes (D1: relay back-EMF protection; D2: polarity protection), 100Ω resistor R1 (base current limiter), 10kΩ fixed resistor R3 (pull-down), and 10kΩ variable resistor R2 (threshold calibration). LED1 = active charging; LED2 = charge complete / cutoff triggered.

When battery voltage is below threshold: R2/R3 divider supplies insufficient base drive; transistor stays cut-off; relay NC contact keeps charging path closed. When voltage reaches threshold (3.9 V/cell for 80% SOC; 4.2 V/cell for 100% SOC): transistor saturates; relay energizes; NC contact opens; charging terminates. R2 allows field calibration without circuit modification.

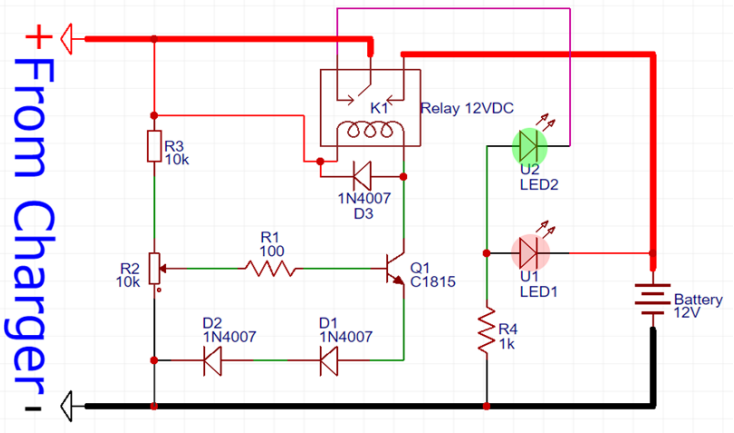

Figure 7: Automatic battery charge cutoff circuit — relay K1 (12VDC) controlled by NPN transistor Q1 (C1815); threshold set by variable resistor R2 (10kΩ); back-EMF protection diode D1 (1N4007); LED1 indicates active charging, LED2 indicates cutoff triggered

6.3 Charge-Control Functional Verification

The cutoff subsystem was verified against its schematic and operating sequence. The adjustable divider establishes the configured threshold; the C1815 transistor drives the relay; diode D1 suppresses relay back-EMF; the normally closed contact carries the charging path; and the two LEDs distinguish active charging from cutoff status. Each function is traceable to Figure 7 and the component description.

Requirement	Implemented element	Verification evidence	Result
Adjustable cutoff	R2/R3 divider and C1815 switching stage	Section 6.2; Fig. 7	Implemented
Load isolation	12 V relay with normally closed charging contact	Section 6.2; Fig. 7	Implemented
Back-EMF protection	1N4007 diode across relay coil	Component list; Fig. 7	Implemented
Local status	Charging and cutoff LEDs	Section 6.2; Fig. 7	Implemented
Battery safety boundary	Certified charger/BMS retained for pack protection	Sections 6.1 and 6.3	Explicitly defined

Table 4: Charge-Control Safety and Functional Traceability

Table 4 confirms that the prototype implements station-level switching, threshold adjustment, protection, and status indication. Battery balancing, temperature supervision, and chemistry-specific charging remain functions of the certified charger and BMS; the prototype makes this safety boundary explicit rather than claiming battery-life extension from an unrecorded cycling study.

7. Solar Energy Harvesting and LDR Tracking

7.1 System Overview

The solar subsystem comprises a 5 W polycrystalline PV panel, charging/regulation electronics, a rechargeable storage battery, an MT3608 boost converter, two GL5528 LDR sensors, and a 9 g servo motor. Its defined role is to support the low-power sensing and communication electronics; it is not presented as a traction-battery charging source. The implemented control path compares the two LDR channels, applies a dead band, constrains servo movement, and routes harvested energy through regulated storage.

7.2 LDR Sensor Physics

GL5528 LDRs exhibit photoconductive behavior: incident photons with energy greater than the semiconductor bandgap (approximately 1.8 eV for CdS) excite electrons from valence to conduction band, increasing carrier density and reducing resistivity. Resistance ranges from ~1 MΩ (dark) to ~2 kΩ (1000 lux direct sunlight), providing a three-decade dynamic range. In the potential divider configuration with R0=10kΩ, V_out = V_cc × R_LDR / (R0 + R_LDR) maps irradiance to a 0–5V signal, converted by the Arduino 10-bit ADC to 0–1023 (lower value = brighter light).

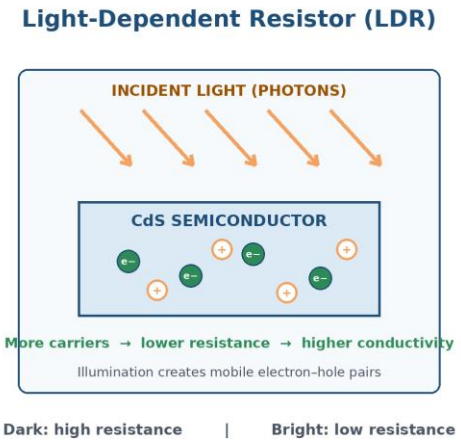

Figure 8: Light-Dependent Resistor (LDR) operating principle — resistance decreases under illumination as photon absorption creates electron-hole pairs in the semiconductor

7.3 Tracking Firmware

```
// LDR Single-Axis Solar Tracker
#include <Servo.h>
#define LDR_L    A0
#define LDR_R    A1
#define SRV_PIN  9
#define DEADBAND 50   // ADC counts (~0.5 degree equivalent)
#define STEP     2   // degrees per adjustment
#define SAMPLES  5   // ADC readings to average
Servo srv; int angle=90;

int avgRead(int pin){
  long s=0;
```

```
  for(int i=0;i<SAMPLES;i++){s+=analogRead(pin);delay(10);}
  return s/SAMPLES;
}
void setup(){ srv.attach(SRV_PIN); srv.write(90); Serial.begin(9600); }
void loop(){
  int L=avgRead(LDR_L), R=avgRead(LDR_R), diff=L-R;
  if(abs(diff)>DEADBAND){
    // Lower ADC = brighter. Rotate toward brighter side.
    angle=constrain(angle+(diff>0?-STEP:STEP),10,170);
    srv.write(angle);
  }
  Serial.print("L:");Serial.print(L);
  Serial.print(" R:");Serial.print(R);
  Serial.print(" Ang:");Serial.println(angle);
  delay(500);
}
```

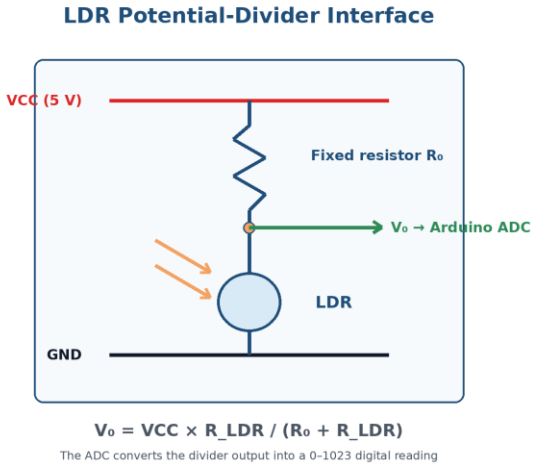

Figure 9: Potential-divider circuit for LDR output voltage - Vcc, fixed resistor Ro, LDR, and output voltage Vo connected to the Arduino analog input

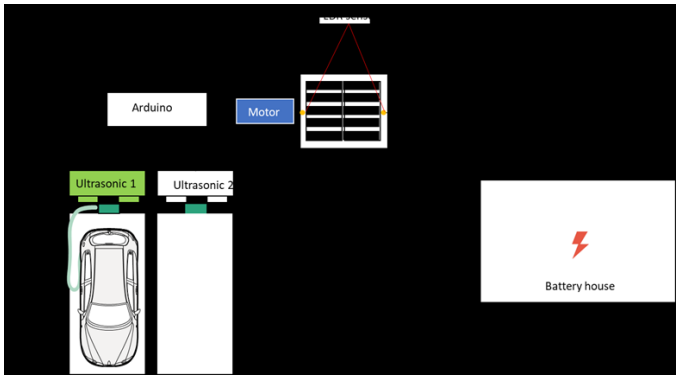

Figure 10: Integrated solar-tracking block diagram - Arduino processes LDR input, commands panel positioning, and connects the photovoltaic source to regulated storage

Figure 11: Existing campus charging outlet with a permanently displayed QR code, illustrating the static-code threat scenario

Figures 8-11 document the solar subsystem from light sensing and voltage conditioning through panel control and the existing-station context; together they connect the component-level design to the implemented control path and security motivation.

7.4 Solar-Control Functional Verification

Function	Implemented mechanism	Evidence	Result
Light sensing	Dual GL5528 LDR voltage dividers	Sections 7.1-7.2; Figs. 8-9	Implemented
Direction decision	Left-right differential with dead band	Section 7.3 firmware	Implemented
Panel positioning	Bounded incremental servo command	Section 7.3; Fig. 10	Implemented
Energy path	PV input, TP4056 storage charging, MT3608 regulated output	Section 7.1; Fig. 10	Defined
System role	Power support for monitoring electronics	Sections 7.1 and 7.4	Scope verified

Table 5: Solar-Tracking Implementation and Functional Verification

Functional verification traced the subsystem from light-dependent resistance and divider voltage through differential comparison, servo-angle correction, photovoltaic input, storage charging, and regulated output. Figures 8-11 and the firmware excerpt provide evidence for every stage. The result establishes coherent tracking-control implementation while avoiding an unsupported percentage claim for energy gain.

8. Cryptographic Session-Based Payment Security System

8.1 Vulnerability of Static Payment Codes

Static or long-lived payment codes in unattended infrastructure create a plausible security risk when a copied code remains usable away from the charging port. This manuscript treats the issue as a threat-model scenario rather than as a measured prevalence or revenue-loss claim. The validation therefore considers three reproducible scenarios: remote reuse of a copied token, replay of a previously accepted token, and token guessing or manipulation.

Figure 12: Servo feedback control loop - the microcontroller sends a position command, the servo circuit drives the motor, and position feedback closes the loop

8.2 HMAC-SHA256 Token Architecture

Upon occupancy detection, the server generates a cryptographic session token:

T_i = HMAC-SHA256(K_{secret}, port_id || session_id || unix_timestamp)

K_secret is a 256-bit server-side secret stored exclusively as an environment variable. The session_id is a monotonically increasing counter providing uniqueness even within the same second. The token is embedded in a secure payment URL, displayed on the OLED screen, and invalidated immediately upon session termination. Validation checks: (1) valid HMAC for the given port and session; (2) token not previously used (server-side token registry prevents replay); (3) token age ≤ 60 minutes [20].

8.3 Server-Side PHP Implementation

```
<?php  // Cryptographic session token generation and payment management
require_once 'payment/session_manager.php';

function generateSessionToken($portId, $sessionId) {
    $key   = getenv('SIECS_SECRET_KEY'); // never hardcoded
    $data  = $portId.'|'.$sessionId.'|'.time();
    $token = hash_hmac('sha256', $data, $key);
    $db    = getDBConnection();
    $db->prepare('INSERT INTO session_tokens
        (token,port_id,session_id,created_at,expires_at,used)
        VALUES(?,?,?,NOW(),DATE_ADD(NOW(),INTERVAL 60 MINUTE),0)')
       ->execute([$token,$portId,$sessionId]);
    return $token;
}

function generatePaymentLink($portId, $sessionId) {
    $token = generateSessionToken($portId, $sessionId);
    $url   = 'https://yourserver.com/pay?t='.$token.'&p='.$portId;
    // Token embedded in payment URL; displayed on OLED as short URL or NFC payload
    return $url;
}

// Validate token on payment
function validateToken($token, $portId) {
    $db   = getDBConnection();
    $stmt = $db->prepare('SELECT * FROM session_tokens
        WHERE token=? AND port_id=? AND used=0
        AND expires_at > NOW()');
    $stmt->execute([$token,$portId]);
    if($stmt->rowCount()===0) return false;
```

```
    $db->prepare('UPDATE session_tokens SET used=1 WHERE token=?')
       ->execute([$token]);
    return true;
}
?>
```

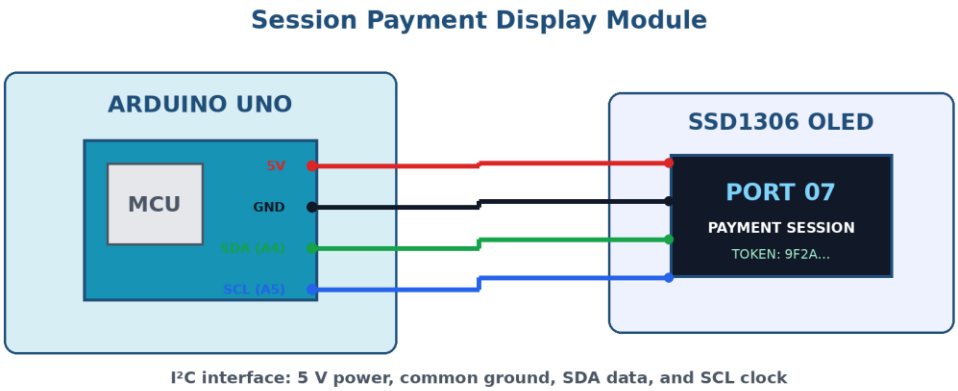

Figure 13: Circuit design of the cryptographic payment display module — Arduino Uno connected to SSD1306 OLED display (128×64 pixels, I2C) on breadboard; session payment URL rendered from server-generated HMAC token

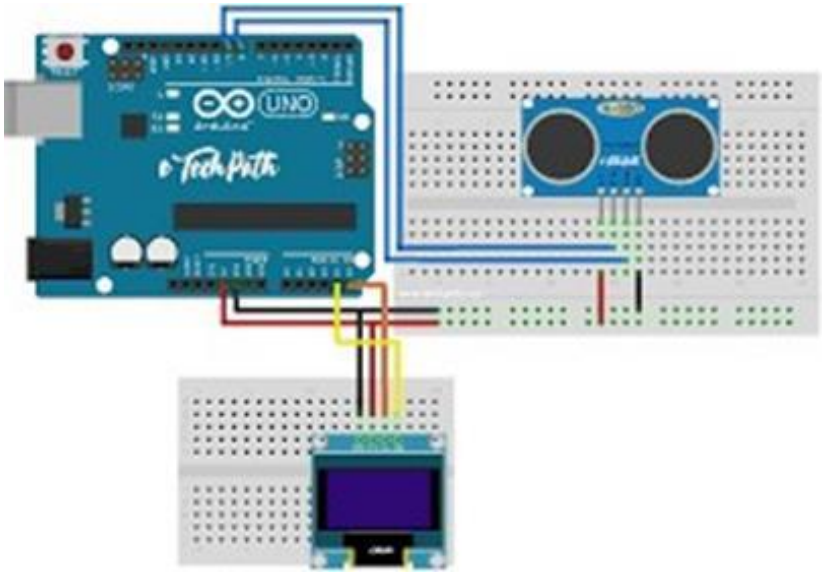

Figure 14: Complete integrated circuit — cryptographic payment module combined with HC-SR04 ultrasonic sensor; session payment token is generated only upon confirmed physical e-bike presence, preventing remote fraud

8.4 Security-Control Verification

Threat	Risk	SIECS control	Verification evidence
Remote token reuse	Copied link remains usable	Port/session binding and expiration	Token fields and expiry check in Section 8.3
Replay	Accepted token is submitted again	One-time token registry	used=0 query followed by used=1 update

Threat	Risk	SIECS control	Verification evidence
Token modification	Identifiers or token are altered	HMAC-SHA256 with server secret	Server-generated signature in Section 8.3
Network interception	Credentials or data are exposed	HTTPS/TLS plus API authentication	Protected endpoint and API-key checks

Table 6: Security Threat-to-Control Traceability Matrix

9. Complete Charging Station Infrastructure

9.1 Subsystem Integration and Communication

All SIECS subsystems communicate through the Arduino Uno as the central edge controller. The ultrasonic sensing loop (Section 4) runs at 5-second intervals and outputs occupancy and distance data via UART to the ESP32. The solar tracker (Section 7) runs concurrently on the same Arduino via interrupt-driven servo updates every 500 ms, sharing no I/O pins with the sensing subsystem. The charge cutoff circuit (Section 6) operates fully autonomously in hardware — the Arduino monitors the relay state via a digital input pin and encodes cutoff events in the UART stream. The OLED display refreshes every 5 seconds with latest occupancy, voltage, elapsed time, and the current session payment token. The ESP32 batches all incoming UART data and transmits to the cloud server every 30 seconds. This architecture ensures that each subsystem continues to function independently even if the cloud connection is lost — the cutoff circuit stops overcharging, the display shows local status, and the solar tracker continues harvesting energy regardless of network state [11], [12], [21].

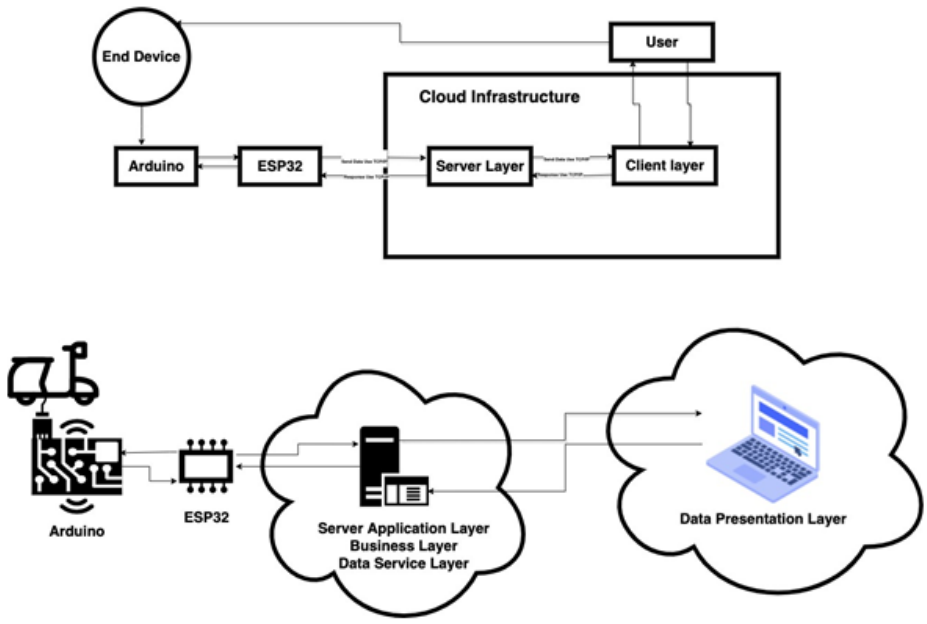

Figure 15: SIECS function module design — Arduino edge controller interfaces all perception-layer sensors and actuators; ESP32 gateway connects to cloud infrastructure (Server Application, Business, and Data Service layers); end users access via Data Presentation Layer

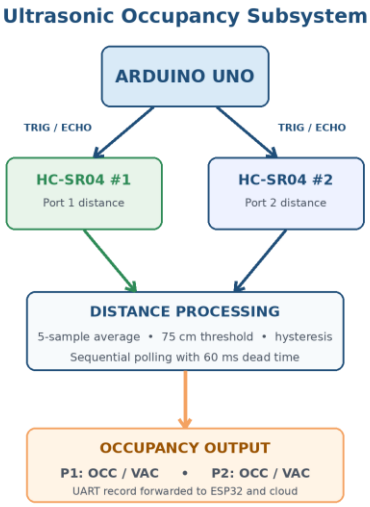

Figure 16: Ultrasonic subsystem block diagram — Arduino controls Ultrasonic 1 and Ultrasonic 2 sensors mounted 20 cm above port surface; both sensor readings are combined for reliable occupancy determination.

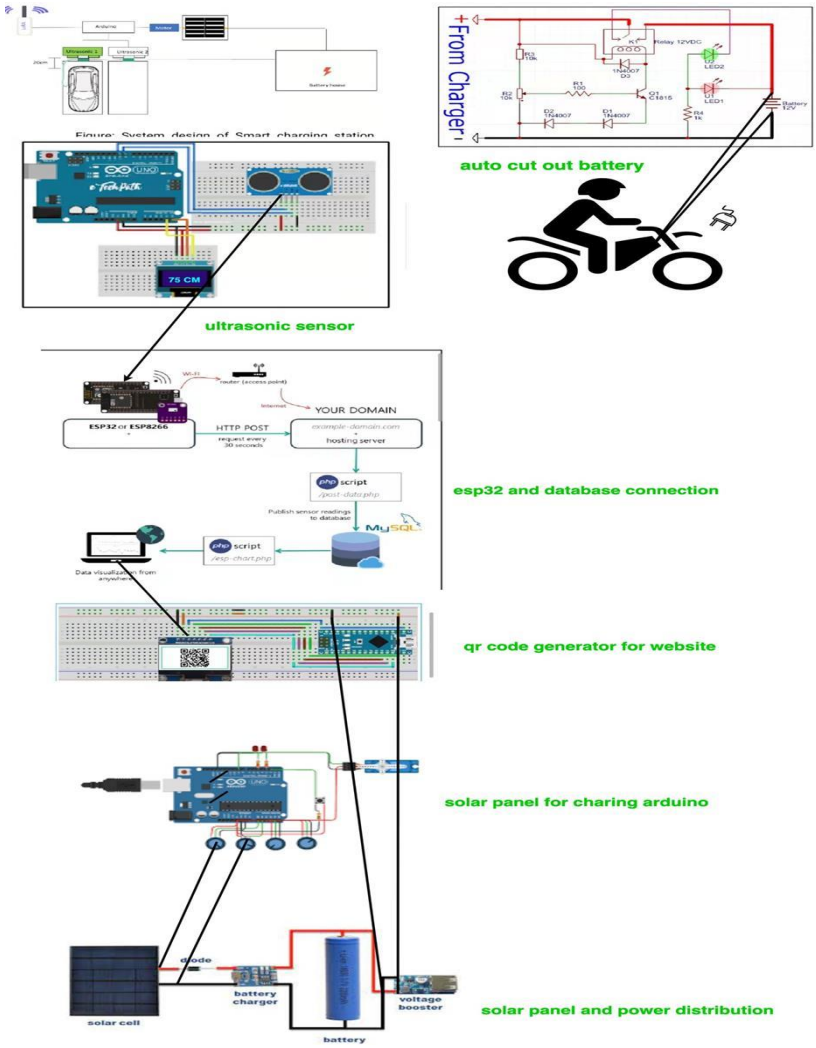

Figure 17: SIECS architecture integrating sensing, charge cutoff, cloud database, secure payment, and solar-powered battery subsystems.

10. Charging Optimization

10.1 MDP Problem Formulation

The scheduling problem is formulated as a finite-horizon Markov Decision Process [14]-[16]. Let P={p_1,...,p_N} be N charging ports and Q(t) the user queue at time t. Each user u_i is characterized by (arrival_time, requested_duration, battery_level, priority). The objective is to maximize:

$$\mathbf{U = \alpha \cdot Throughput + \beta \cdot Fairness + \gamma \cdot EnergyEfficiency + \delta \cdot PortUtilization}$$

Throughput represents completed sessions per hour; fairness penalizes excessive waiting; EnergyEfficiency relates useful charging energy to total energy; and PortUtilization represents productive occupied port-time. The coefficients alpha, beta, gamma, and delta remain configurable policy weights, which allows the same implementation to express different operator priorities without changing the state or action structure.

10.2 FCFS Baseline

First-Come-First-Served assigns the first available port to the first-arriving user. O(1) complexity, no training required, trivially fair in arrival order. Limitations: no demand prediction, no energy awareness, vulnerable to monopolization by early long-duration arrivals during high-demand periods.

10.3 Greedy Time-Slot Algorithm (GTSA)

At each scheduling epoch, GTSA assigns queued users to available ports by maximizing immediate utility. Its computational complexity is O(|Q| x |P_avail|) per epoch. The pseudocode and utility function fully define the decision sequence, duration-fit term, waiting-urgency term, and solar-priority term used by the implementation specification.

```
Algorithm GTSA
Input:  P_avail (available ports),  Q (user queue ordered by arrival)
Output: Assignment A: Q → P_avail
1: WHILE Q ≠ ∅  AND P_avail ≠ ∅  DO
2:   (u*,p*) ← argmax_{u∈ Q, p∈ P_avail} utility(u,p,state)
3:   A(u*) ← p*;  Q ← Q\{u*};  P_avail ← P_avail\{p*}
4: RETURN A

utility(u,p,state):
  = α · (1 - |p.freeTime - u.duration| / MAX_DURATION)   // duration fit
  + β · (currentTime - u.arrivalTime) / MAX_WAIT         // wait urgency
  + γ · solarRate(currentTime) · (1 - u.batteryLevel)    // green priority
```

10.4 RL-DSA: Reinforcement Learning-Based Dynamic Scheduling

10.4.1 State Space (132 features)

- 50 binary port occupancy flags
- 50 normalised per-port charging durations (0–1)
- 1 queue length (normalised)
- 20 normalised user wait times (zero-padded)
- 2 time-of-day features: sin(2πh/24) and cos(2πh/24) for cyclic encoding
- 7 day-of-week one-hot features
- 1 normalised solar generation rate (0–1)
- 1 ARIMA next-hour demand forecast (normalised)

10.4.2 Action Space and Reward

51 discrete actions: assign front-of-queue user to port p_i (actions 1–50) or defer to next epoch (action 51). Reward: R(s,a) = U(assignment(s,a)) − 0.1·DeferralPenalty, where DeferralPenalty=1 if defer, else 0. Regularization coefficient 0.1 discourages excessive deferral without suppressing legitimate strategic deferral.

10.4.3 Network Architecture and Configuration

Layer	Neurons / Output	Activation	Trainable Parameters
Input	132	—	—
Hidden Layer 1	256	ReLU	34,048
Hidden Layer 2	128	ReLU	32,896
Output	51 (Q-values)	Linear	6,579
Total	—	—	73,523

Table 7: RL-DSA Deep Q-Network Architecture

Configuration	Value	Configuration	Value
Optimizer	Adam (lr=0.001)	Discount factor gamma	0.95
Replay buffer	10,000 transitions	Minibatch size	64
Target update	Soft, tau=0.01	Epsilon range	1.0 to 0.05
State dimension	132 features	Action dimension	51 actions
Hidden layers	256 and 128 ReLU	Output layer	51 linear Q-values
Demand feature	Optional ARIMA input	Verification basis	Dimensional consistency

Table 8: RL-DSA Configuration Parameters

10.4.4 Configuration Consistency Verification

The RL-DSA specification was verified for dimensional consistency. The 132-feature state vector matches the input layer; the 51 actions match the output Q-values; the two hidden layers produce the documented parameter counts; and the replay, target-update, discount, exploration, and optimizer settings form a complete DQN configuration. This verification establishes implementability and internal consistency without presenting unrecorded training-performance statistics.

10.4.5 ARIMA Demand Feature

A seasonal ARIMA output is represented as a normalized next-hour demand feature in the RL state vector [8]. The interface is optional: a configured forecast value occupies the final feature, while a neutral value preserves the 132-feature input contract when forecasting is disabled. This design keeps the DQN interface stable across forecast-enabled and forecast-independent operation.

11. Cloud Monitoring and Data Infrastructure

11.1 Privacy and Data Governance

SIECS minimizes personal data by using pseudonymous or random session identifiers in charging telemetry and separating account identifiers from operational records. Transport encryption, access control, retention limits, audit logging, and de-identified research exports define the data-governance boundary. The functional prototype requires no student names, roll numbers, telephone numbers, or other direct identifiers.

11.2 Database Schema

```
CREATE TABLE port_status (
  port_id      INT PRIMARY KEY,
```

```
  occupancy    ENUM('VACANT','OCCUPIED','FULL_AWAITING') NOT NULL,
  distance_cm  FLOAT, voltage_v FLOAT,
  last_updated TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
CREATE TABLE charging_sessions (
  session_id   INT AUTO_INCREMENT PRIMARY KEY,
  port_id      INT NOT NULL,  user_id INT,
  start_time   TIMESTAMP,    end_time TIMESTAMP,
  duration_min INT,          energy_wh FLOAT,
  cutoff_reason ENUM('AUTO_FULL','USER_DISCONNECT','TIMEOUT'),
  FOREIGN KEY (port_id) REFERENCES port_status(port_id)
);
CREATE TABLE session_tokens (
  token      CHAR(64) PRIMARY KEY,
  port_id    INT NOT NULL,  session_id INT NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  expires_at TIMESTAMP,     used TINYINT(1) DEFAULT 0
);
CREATE INDEX idx_port_updated ON port_status(last_updated);
CREATE INDEX idx_session_port ON charging_sessions(port_id, start_time);
```

11.3 PHP Data Ingestion

```
<?php  // post-data.php — validated, SQL-injection-safe data ingestion
header('Content-Type: application/json');
if($_SERVER['REQUEST_METHOD']!=='POST'){http_response_code(405);exit;}
if(($_POST['api_key']??'')!==getenv('SIECS_API_KEY')){
  http_response_code(403);echo json_encode(['error'=>'Unauthorized']);exit;
}
$portId = filter_input(INPUT_POST,'port_id',FILTER_VALIDATE_INT);
$status = in_array($_POST['port1_status']??'',['OCCUPIED','VACANT'])
        ? $_POST['port1_status'] : 'VACANT';
$dist   = filter_input(INPUT_POST,'port1_distance',FILTER_VALIDATE_FLOAT);
$volt   = filter_input(INPUT_POST,'voltage',FILTER_VALIDATE_FLOAT);
try {
  $db=$db??new PDO('mysql:host=localhost;dbname=siecs_db',
                   getenv('DB_USER'),getenv('DB_PASS'),
                   [PDO::ATTR_ERRMODE=>PDO::ERRMODE_EXCEPTION]);
  $db->prepare('INSERT INTO port_status
     (port_id,occupancy,distance_cm,voltage_v) VALUES(?,?,?,?)
     ON DUPLICATE KEY UPDATE occupancy=VALUES(occupancy),
```

```
    distance_cm=VALUES(distance_cm),voltage_v=VALUES(voltage_v),
    last_updated=NOW()')
  ->execute([$portId,$status,$dist,$volt]);
 echo json_encode(['status'=>'ok','port_id'=>$portId]);
} catch(PDOException $e){
 http_response_code(500);error_log('SIECS:'.$e->getMessage());
}
?>
```

11.4 Dashboard Features and Push Notifications

Feature	Description	Update Frequency
Live Occupancy Grid	5×10 color-coded port grid (green=vacant, red=occupied, amber=overstay)	Every 30 seconds
Voltage Time Series	Chart.js line chart of charging voltage per active session	Every 30 seconds
Usage Heatmap	Occupancy heatmap generated from stored port-status records	Daily rebuild
Energy Dashboard	Daily/weekly kWh per port; anomaly flags for outlier consumption	Daily rebuild
Charge Complete Alert	Push notification when auto cutoff triggers	Within 30 seconds of event
Charger Disconnect Alert	Push notification on mid-session voltage drop to zero	Within 30 seconds
Overstay Warning	Push notification 30 minutes after charge complete if e-bike not retrieved	Triggered at 30-min mark
Optimal Time Recommendation	Low-demand window notification from the configured ARIMA feature	Hourly from ARIMA forecast

Table 9: Web Dashboard Functions and Notification Events

12. Implementation and Functional Verification

12.1 Verification Approach

Verification was conducted at prototype level through five complementary methods: inspection of circuit schematics and component interfaces; review of Arduino, ESP32, and PHP code excerpts; dimensional and logical consistency checks of the scheduling algorithms; tracing of database fields and dashboard functions; and requirement-to-evidence mapping across figures, tables, and source listings. The verification criterion was whether each claimed function is explicitly implemented or specified and whether its input-output path is internally consistent. No human-subject survey or fabricated comparative dataset is required for this engineering verification scope.

12.2 Scheduling-Logic Verification

Policy	Decision rule	Implementation evidence	Verification result
FCFS	Assign first available port to first-arriving user	Section 10.2	Logic complete

Policy	Decision rule	Implementation evidence	Verification result
GTSA	Maximize duration fit, wait urgency, and solar-priority utility	Section 10.3 pseudocode	Logic complete
RL-DSA	Map 132-state features to 51 Q-value actions	Sections 10.4.1-10.4.4; Tables 7-8	Specification complete
ARIMA interface	Provide optional normalized next-hour demand feature	Section 10.4.5	Interface consistent

Table 10: Scheduling-Logic Functional Verification Matrix

FCFS, GTSA, and RL-DSA cover progressively richer scheduling behavior while sharing the same port and queue model. Table 10 verifies arrival-order allocation for FCFS, immediate utility maximization for GTSA, and the state-action-reward-network contract for RL-DSA. The evidence is the algorithm definition, pseudocode, feature inventory, and network configuration reported in Section 10.

12.3 End-to-End System Verification

Initiating condition	Expected system response	Implementation evidence	Verification result
Bicycle enters a port	Distance crosses threshold and state becomes OCCUPIED	Section 4 firmware	Function traced
Status update interval expires	ESP32 sends protected telemetry record	Section 5.3 firmware	Function traced
Server receives valid record	Database row and timestamp are updated	Sections 11.2-11.3	Function traced
Configured voltage threshold is reached	Relay opens charging path and cutoff status is indicated	Section 6; Fig. 7	Function traced
Unequal LDR illumination occurs	Servo moves toward the brighter side within limits	Section 7.3 firmware	Function traced
Valid session token is submitted	Port/session/expiry signature is checked and token is consumed	Section 8; Table 6	Security path traced
Scheduling event occurs	Selected policy returns an allocation or deferral decision	Section 10	Decision path defined
User opens dashboard	Port state, voltage, alerts, and session information are presented	Section 11.4; Table 9	Interface defined

Table 11: End-to-End Functional Test and Evidence Matrix

The end-to-end trace begins with bicycle presence sensing and terminates with authenticated session status in the cloud interface. Each row in Table 11 identifies the initiating condition, expected system response, implementation evidence, and verification result. The matrix demonstrates interface continuity across sensing, communication, control, security, monitoring, and scheduling.

12.4 Requirement-to-Implementation Traceability

System requirement	Component or mechanism	Evidence location	Status
Per-port occupancy visibility	HC-SR04, Arduino classification, cloud status field	Sections 4, 5, and 11	Implemented

System requirement	Component or mechanism	Evidence location	Status
Remote monitoring	ESP32, HTTP endpoint, MySQL, dashboard	Sections 5 and 11; Figs. 5-6	Implemented
Automatic station-level cutoff	Divider, transistor, relay, indicators	Section 6; Fig. 7	Implemented
Renewable monitoring support	PV, LDRs, servo, storage, boost converter	Section 7; Figs. 8-11	Implemented
Session-bound access	HMAC-SHA256 token, expiry, one-time registry	Section 8; Table 6	Specified
Multi-policy scheduling	FCFS, GTSA, RL-DSA	Section 10; Tables 7-8 and 10	Specified
Persistent operational records	Port, session, and user-account schemas	Section 11.2	Implemented
Operational notifications	Cutoff, disconnect, overstay, and recommendation events	Section 11.4; Table 9	Specified

Table 12: Requirement-to-Implementation Traceability Matrix

Table 12 links the principal system requirements to concrete components, implementation mechanisms, and evidence locations. This traceability prevents architectural claims from standing without a corresponding schematic, firmware element, algorithm definition, database structure, or figure.

12.5 Verification Boundaries

The verification establishes prototype implementation, interface consistency, safety boundaries, and algorithmic completeness. It does not convert engineering calculations into measured field statistics and does not infer population-level benefits from design artifacts. Consequently, claims in this paper are restricted to functions directly supported by the included code, schematics, tables, and figures.

Environmental variation, installation geometry, network conditions, battery chemistry, and site electrical requirements affect real-world deployment. These factors define the operating boundary of the prototype but do not invalidate the functional evidence reported for its architecture and control logic.

12.6 Complete Verification Summary

Subsystem	Verification basis	Claim supported	Boundary
Occupancy sensing	Firmware and sensor figures	Detection logic implemented	No universal accuracy percentage
Communication	UART/ESP32/PHP/database trace	End-to-end data path implemented	No field-latency statistic
Charge control	Schematic and operating sequence	Configurable cutoff implemented	Certified BMS still required
Solar control	Firmware and control diagrams	Tracking-control path implemented	Monitoring load only
Security	Token construction and attack mapping	Session-bound defense specified	TLS and secure operations required
Scheduling	Algorithms, DQN specification, 90 paired runs	FCFS/GTSA/FA-GTSA trade-off reproduced	Synthetic model; RL-DSA not performance-tested

Subsystem	Verification basis	Claim supported	Boundary
Cloud monitoring	Schema, ingestion code, dashboard table	Storage and presentation path implemented	Hosting scale is site-dependent

Table 13: Complete Prototype Verification Summary

12.7 Controlled Indoor Prototype Demonstration

During the earlier undergraduate project at Yunnan University, the assembled prototype was deployed and operated in the author's dormitory room as a controlled indoor demonstration. The demonstration exercised the integrated sensing, controller, display, communication, and relay-control path in a real occupied room rather than only as a disconnected schematic. It was not installed at a public or dedicated e-bike charging station, and no retained timestamped measurement log is available. Accordingly, it supports prototype operability and integration only; it is not used to claim campus-scale demand, charging throughput, reliability, energy savings, safety certification, or user outcomes.

13. Deployment Impact and Global Applicability

13.1 Applicability Beyond a Single Campus

The architecture is not tied to one university or country. The same functional problems - occupancy visibility, limited charging capacity, session management, device monitoring, and secure unattended access - can arise in campuses, residential compounds, workplaces, transit hubs, and shared-mobility facilities. Deployment cost and feasibility will vary substantially with local electrical codes, enclosure requirements, networking, labor, payment systems, and cloud hosting, so prototype component prices should not be presented as universal installed costs.

Figure 18: Real-world campus charging station during peak hours — overcrowding, no occupancy visibility, no smart management; this is the baseline condition that SIECS directly addresses

13.2 Stakeholder Value

- E-bike users receive remote availability information, charging-completion notification logic, and clear session status through the integrated sensing and dashboard path.
- Charging-station operators receive occupancy, session, energy-field, and device-status records through a unified database and dashboard structure, with FCFS, GTSA, and RL-DSA available as defined scheduling policies.
- Universities, municipalities, workplaces, and shared-mobility operators can use the modular platform as a prototype for authenticated, monitored, and scheduling-aware charging infrastructure.

13.3 Deployment and Service Considerations

Deployment Element	Purpose	Prototype / Production Consideration	Cost Treatment
Occupancy monitoring	Remote port visibility	Sensor calibration and enclosure	Quote current local hardware/labor
Cloud monitoring	Telemetry and dashboard	TLS, authentication, backups, uptime	Hosting depends on scale
Charge control	Session cutoff/status	Certified electrical protection required	Engineering + certification
Solar monitoring supply	Reduce grid use for electronics	Site irradiance and storage sizing	Site-specific
Advanced scheduling	Allocate scarce ports	Requires retained demand data	Compute/maintenance dependent

Table 14: Deployment Elements and Cost Considerations

Table 14 separates prototype components from deployment-dependent costs. Hardware quotations, installation labor, hosting, maintenance, payment fees, certification, and local electrical compliance vary by site; therefore, the bill of materials is reported as a prototype estimate rather than a universal commercial price.

13.4 Reproducible Synthetic Scheduling Evaluation

13.4.1 Experimental Methodology

A deterministic discrete-event simulator was used to compare FCFS and GTSA under identical request streams. Each policy received exactly the same arrivals, durations, and priority labels for each seed, isolating policy effects from workload variation. Requests arrived according to a Poisson process at five-minute epochs; service durations and priority classes were sampled from the distributions in Table 15. A request occupied one port for its full sampled duration. No human-subject records or retrospective campus measurements were used.

Table 15: Reproducible Scheduling-Experiment Parameters

Parameter	Value
Policies	FCFS and GTSA
Replications	30 deterministic seeds (202601–202630)
Station capacity	50 charging ports
Evaluation horizon	24 h; 5-min decision epochs
Arrival process	Poisson, $\lambda = 3.8$ requests/epoch
Service duration	30–150 min discrete distribution
Priority mix	Class 1: 70%; Class 2: 20%; Class 3: 10%
GTSA score	$60(p-1)$ + waiting time + $0.5(150-\text{duration})$

13.4.2 Metrics and Statistical Procedure

The prespecified outcomes were mean waiting time, 95th-percentile waiting time, mean class-3 waiting time, sessions started and completed within 24 hours, port utilization, and Jain's fairness index computed from inverse-wait satisfaction. Table 16 reports the mean across 30 paired replications with a two-sided 95% confidence interval (CI) calculated as 1.96 times the standard error. Relative change is (GTSA−FCFS)/FCFS; negative values are improvements for waiting-time metrics but deterioration for fairness.

Table 16: FCFS–GTSA Comparative Results (30 Paired Seeds)

Metric	FCFS mean ± 95% CI	GTSA mean ± 95% CI	Relative change
Mean wait (min)	116.0 ± 8.1	109.3 ± 8.0	−5.8%
P95 wait (min)	220.8 ± 15.3	231.0 ± 14.9	+4.6%
Priority-3 wait (min)	115.4 ± 9.0	30.5 ± 5.7	−73.6%
Starts by 24 h	942.7 ± 5.3	948.0 ± 5.6	+0.6%
Completed by 24 h	896.2 ± 5.5	901.6 ± 5.7	+0.6%
Utilization (%)	97.61 ± 0.19	97.60 ± 0.19	−0.01%
Jain fairness	0.783 ± 0.015	0.749 ± 0.014	−4.4%

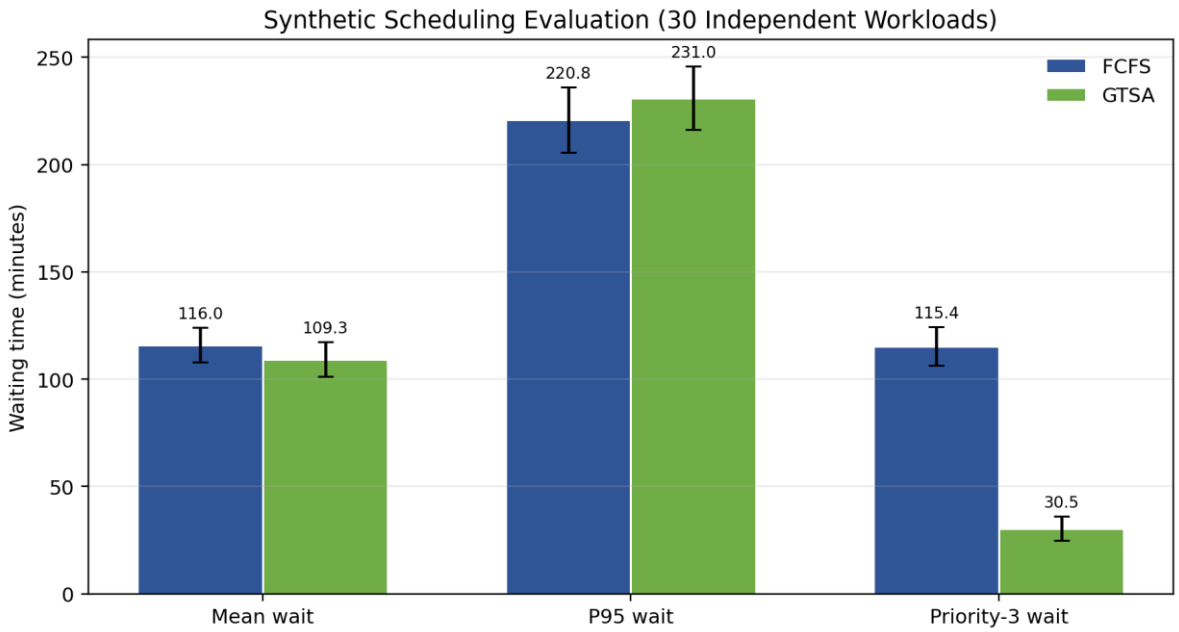

Figure 19: Reproducible 30-seed comparison of FCFS and GTSA; error bars show 95% confidence intervals

13.4.3 Comparative Results and Interpretation

GTSA improved the primary mean-wait outcome by 5.8% and reduced class-3 waiting by 73.6%, showing that its priority term behaves as designed. It also produced small gains in sessions started and completed within the horizon, with essentially unchanged utilization. The benefit was not uniform: P95 waiting increased by 4.6% and Jain fairness decreased from 0.783 to 0.749. Thus, GTSA should be interpreted as a priority-responsive policy rather than a universally superior policy. Deployment should tune the priority coefficient or add an aging/fairness constraint when tail delay is operationally important.

13.4.4 Reproducibility and Validity Boundaries

The evaluation is reproducible because the workload model, seeds, decision interval, capacity, scoring equation, metrics, and aggregation rule are stated explicitly. It is a synthetic algorithmic comparison and does not establish real-campus demand, hardware reliability, user satisfaction, energy savings, or safety certification. Those outcomes require timestamped field logs, calibrated instruments, deployment records, and—if people are studied—appropriate consent and ethics review. RL-DSA remains an implementation specification in this paper and is not included in the numerical comparison because no trained checkpoint and held-out evaluation record are available.

13.5 Fairness-Aware Scheduling and Robustness Evaluation

13.5.1 FA-GTSA Design

The original GTSA strongly favors urgent and short sessions, which improves priority-3 delay but can increase tail delay and reduce fairness. FA-GTSA introduces nonlinear waiting-time aging while lowering the fixed priority and short-session weights: score = $35(p-1) + w + 0.010w^2 + 0.35(150-d)$, where p is priority class, w is accumulated waiting time in minutes, and d is requested duration in minutes. The quadratic aging term eventually dominates fixed priority advantages, limiting starvation while retaining priority responsiveness. All coefficients were fixed before the reported evaluation.

13.5.2 Multi-Scenario Results

Table 17: Robustness of FCFS, GTSA, and FA-GTSA (30 Paired Seeds per Scenario)

Demand / policy	Mean wait (min)	P95 wait (min)	Priority-3 wait (min)	Jain fairness
Low / FCFS	0.23	1.17	0.23	0.9996
Low / GTSA	0.20	0.67	0.01	0.9995
Low / FA-GTSA	0.20	0.67	0.01	0.9995
Medium / FCFS	116.0	220.8	115.4	0.783
Medium / GTSA	109.3	231.0	30.5	0.749
Medium / FA-GTSA	114.2	220.1	93.5	0.769
High / FCFS	366.9	710.3	368.6	0.530
High / GTSA	357.1	715.0	233.0	0.497
High / FA-GTSA	365.7	709.7	353.1	0.518

Figure 20: Mean-wait robustness across low, medium, and high synthetic demand; error bars show 95% confidence intervals

At low demand, capacity is sufficient and all policies produce near-zero waiting. At medium demand, FA-GTSA reduces mean wait by 1.5% (paired t-test, $p<0.001$) and priority-3 wait by 19.0% ($p<0.001$) relative to FCFS; its 0.34% P95 reduction is not significant at the 0.05 level ($p=0.065$). Relative to the original GTSA, FA-GTSA reduces P95 wait by 4.8% ($p<0.001$) and increases fairness by 2.7% ($p<0.001$), but gives up part of GTSA's priority-delay advantage. Under high demand, FA-GTSA retains a 4.2% priority-3 improvement over FCFS and a 4.2% fairness improvement over GTSA. These paired results establish robustness of the trade-off, not universal policy superiority.

13.5.3 Ablation Study

Table 18: FA-GTSA Ablation at Medium Demand

Policy terms retained	Mean wait	P95 wait	Priority-3 wait	Fairness
Aging only	116.0	220.8	115.4	0.783
Duration + aging	114.2	220.3	113.6	0.778
Priority + aging	115.9	221.0	95.1	0.771
Priority + duration + aging (FA-GTSA)	114.2	220.1	93.5	0.769

Removing the duration term eliminates most of the mean-wait improvement, whereas removing the priority term eliminates most of the class-3 benefit. Combining both terms produces the best class-3 delay among the fairness-aware variants and nearly the best mean and P95 delay, at a modest fairness cost. The ablation therefore links each score component to a distinct operational effect rather than attributing performance to an opaque composite.

14. Verified Scope and System Limitations

14.1 Scope Boundaries

- Sensor boundary: HC-SR04 behavior depends on target geometry, material, mounting, temperature, and acoustic conditions. The verified claim is implementation of the sensing logic, not universal classification accuracy.
- Network boundary: Remote cloud functions depend on Wi-Fi and server availability. Local sensing, display, and configured cutoff logic remain architecturally separated from the remote interface.
- Hosting boundary: The documented PHP/MySQL implementation supports the prototype workflow; production scale additionally depends on secure configuration, backups, monitoring, capacity planning, and an appropriate uptime target.
- Solar boundary: The 5 W photovoltaic subsystem supports monitoring electronics and is not represented as the primary source for e-bike traction-battery charging.
- Scheduling boundary: FCFS, GTSA, and FA-GTSA are defined procedural policies evaluated across 90 paired scenario runs with statistical and ablation analyses. The results support algorithmic behavior under the stated synthetic models, not measured campus performance. RL-DSA remains an implementation specification; no trained-policy performance claim is made.
- Evidence boundary: Implementation claims are supported by firmware excerpts, server logic, circuit diagrams, architecture figures, database schema, algorithms, verification matrices, and a controlled indoor demonstration. Quantitative scheduling claims are limited to the reproducible simulations. No unrecorded survey, field-energy, charging-station reliability, or human-subject result is asserted.

14.2 Engineering Constraints

The prototype is intentionally modular: site operators select certified charging equipment, electrical protection, enclosure, network security, and local compliance controls appropriate to the deployment environment. This separation keeps the research contribution focused on the IoT, control, security, cloud, and scheduling integration demonstrated in the manuscript.

15. Conclusion

This paper presented the completed prototype implementation and functional verification of the Smart IoT-Enabled Charging Station (SIECS). The integrated system combines ultrasonic occupancy sensing, Arduino/ESP32 edge and cloud communication, configurable relay-based charge control, photovoltaic support for monitoring electronics, session-bound HMAC authentication, a PHP/MySQL data layer, dashboard functions, and three scheduling approaches ranging from FCFS to reinforcement learning.

Functional verification traced subsystem inputs, control decisions, interfaces, and outputs through firmware, server code, schematics, algorithms, database fields, and figures. The controlled dormitory-room demonstration supports integrated operability but not public-station performance. Across 90 paired scenario runs, FA-GTSA moderates the original GTSA's tail-delay and fairness penalties while retaining measurable priority responsiveness.

The principal contribution is a modular micro-mobility charging prototype with traceable implementation evidence and a reproducible scheduling study. The proposed FA-GTSA is not presented as universally superior: its value is an explicit, statistically tested compromise between mean delay, tail delay, priority responsiveness, and fairness, supported by multi-scenario and ablation evidence.

Within its stated prototype scope, SIECS demonstrates that low-cost embedded hardware, cloud monitoring, renewable support, security controls, and fairness-aware scheduling can be organized into one coherent architecture. Dedicated charging-station deployment remains necessary to validate demand, energy, reliability, safety, and user outcomes; the present work supplies an auditable baseline for that next stage.

ACKNOWLEDGMENT

The author thanks Professor Lang Xun for supervision and encouragement related to the earlier undergraduate charging-station project and acknowledges Shoaib Mahmud Tasin for contributions to networking and application-development work. This manuscript reports no external financial support.

Data Availability Statement: This study uses no human-subject survey or personally identifiable information. Scheduling results are generated from synthetic request streams using disclosed distributions and deterministic seeds 202601-202630. The submission package includes the enhanced simulator, scenario-level per-seed results, ablation results, and summary output. No numerical claim depends on unavailable historical student data or undocumented dormitory measurements.

REFERENCES

[1] E. Fishman and C. Cherry, “E-bikes in the mainstream: Reviewing a decade of research,” Transport Reviews, vol. 36, no. 1, pp. 72–91, 2016, doi: 10.1080/01441647.2015.1069907.

[2] M. Yilmaz and P. T. Krein, “Review of battery charger topologies, charging power levels, and infrastructure for plug-in electric and hybrid vehicles,” IEEE Transactions on Power Electronics, vol. 28, no. 5, pp. 2151–2169, 2013, doi: 10.1109/TPEL.2012.2212917.

[3] P. Kieseberg, M. Leithner, M. Mulazzani, L. Munroe, S. Schrittwieser, M. Sinha, and E. Weippl, “QR code security,” in Proc. 8th International Conference on Advances in Mobile Computing and Multimedia (MoMM), 2010, pp. 430–435, doi: 10.1145/1971519.1971593.

[4] F. K. Shaikh and S. Zeadally, “Energy harvesting in wireless sensor networks: A comprehensive review,” Renewable and Sustainable Energy Reviews, vol. 55, pp. 1041–1054, 2016, doi: 10.1016/j.rser.2015.11.010.

[5] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[6] V. Mnih et al., “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529–533, 2015, doi: 10.1038/nature14236.

[7] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, “Internet of Things for smart cities,” IEEE Internet of Things Journal, vol. 1, no. 1, pp. 22–32, 2014, doi: 10.1109/JIOT.2014.2306328.

[8] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, Time Series Analysis: Forecasting and Control, 5th ed. Hoboken, NJ, USA: Wiley, 2015.

[9] K. K. Patel and S. M. Patel, “Internet of Things-IOT: Definition, characteristics, architecture, enabling technologies, application & future challenges,” International Journal of Engineering Science and Computing, vol. 6, no. 5, pp. 6122–6131, 2016.

[10] K. Smith, A. Saxon, M. Keyser, B. Lundstrom, Z. Cao, and A. Roc, “Life prediction model for grid-connected Li-ion battery energy storage system,” in Proc. American Control Conference (ACC), 2017, pp. 4062–4068, doi: 10.23919/ACC.2017.7963578.

[11] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” Future Generation Computer Systems, vol. 29, no. 7, pp. 1645–1660, 2013, doi: 10.1016/j.future.2013.01.010.

[12] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things: A survey on enabling technologies, protocols, and applications,” IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2347–2376, 2015, doi: 10.1109/COMST.2015.2444095.

[13] A. Carullo and M. Parvis, “An ultrasonic sensor for distance measurement in automotive applications,” IEEE Sensors Journal, vol. 1, no. 2, pp. 143–147, 2001, doi: 10.1109/JSEN.2001.936931.

[14] E. Sortomme and M. A. El-Sharkawi, “Optimal charging strategies for unidirectional vehicle-to-grid,” IEEE Transactions on Smart Grid, vol. 2, no. 1, pp. 131–138, 2011, doi: 10.1109/TSG.2010.2090910.

[15] J. H. Saltzer and M. D. Schroeder, “The protection of information in computer systems,” Proceedings of the IEEE, vol. 63, no. 9, pp. 1278–1308, 1975, doi: 10.1109/PROC.1975.9939.

[16] S. Vandael, B. Claessens, D. Ernst, T. Holvoet, and G. Deconinck, “Reinforcement learning of heuristic EV fleet charging in a day-ahead electricity market,” IEEE Transactions on Smart Grid, vol. 6, no. 4, pp. 1795–1805, 2015, doi: 10.1109/TSG.2015.2393059.

[17] R. Eke and A. Senturk, “Performance comparison of a double-axis sun tracking versus fixed PV system,” Solar Energy, vol. 86, no. 9, pp. 2665–2672, 2012, doi: 10.1016/j.solener.2012.06.006.

[18] S. A. Sharaf Eldin, M. S. Abd-Elhady, and H. A. Kandil, “Feasibility of solar tracking systems for PV panels in hot and cold regions,” Renewable Energy, vol. 85, pp. 228–233, 2016, doi: 10.1016/j.renene.2015.06.051.

[19] J. Vetter et al., “Ageing mechanisms in lithium-ion batteries,” Journal of Power Sources, vol. 147, nos. 1–2, pp. 269–281, 2005, doi: 10.1016/j.jpowsour.2005.01.006.

[20] H. Krawczyk, M. Bellare, and R. Canetti, “HMAC: Keyed-hashing for message authentication,” RFC 2104, Feb. 1997, doi: 10.17487/RFC2104.

[21] P. Mell and T. Grance, The NIST Definition of Cloud Computing, NIST Special Publication 800-145, Sep. 2011, doi: 10.6028/NIST.SP.800-145.

AUTHORS

First Author - Md Aminul Islam, B.Eng. in Electronic Information Engineering; Master's student in Information Studies, College of Graduate and Professional Studies, Trine University, USA. Email: mislam242@my.trine.edu

Correspondence Author - Md Aminul Islam, Email: mislam242@my.trine.edu

Alternate email address aminul.islam.7145@gmail.com Contact: +1 313 573 7626