Comprehensive Build Manual: RFID-Based Productivity Tracker (Version 1.0)

Before diving into the technical details, key findings confirm that an ESP32-WROOM-32 coupled with an MFRC522 RFID reader can reliably log tagged activities to Google Sheets via HTTP POST, enforce per-day usage limits using on-board flash (Preferences library) rather than EEPROM, and deliver audio feedback through an active buzzer—all while running from a 5 V USB supply or battery pack with optional deep-sleep for power savings[1] [2] [3]. Careful pin mapping, Wi-Fi reconnection routines, and Apps Script security settings are pivotal to long-term stability[4] [5] [6].

1 — System Overview

The Version 1.0 tracker records each card tap as a structured JSON object (timestamp, UID, activity, status) and appends it to a Google Sheet through a published Apps Script web app[2] [7]. Cards mapped to "Study", "Exercise", etc. increment habit counters, while restricted cards are blocked after one successful log per UTC day using the Preferences library to store last-use dates[8] [9].

2 — Bill of Materials

#	Component	Notes	Typical Cost
1	ESP32-WROOM-32 DevKit	Wi-Fi/BLE MCU, 4 MB flash	₹350
2	MFRC522 RFID reader + 2 MIFARE cards	13.56 MHz SPI interface	₹200
3	Active piezo buzzer	3.3–5 V, ~30 mA peak	₹20
4	Breadboard (400 tie)	rapid prototyping	₹80
5	Jumper wires (M–M)	20 cm assorted	₹50
6	5 V USB wall adapter or power bank	≥500 mA	₹150
7	Optional RGB LED + 220 Ω resistor	visual status	₹15

Total estimated outlay ≈ ₹865 (US $10.5) for core hardware.

3 — Pinout & Wiring

3.1 ESP32 Core Pins

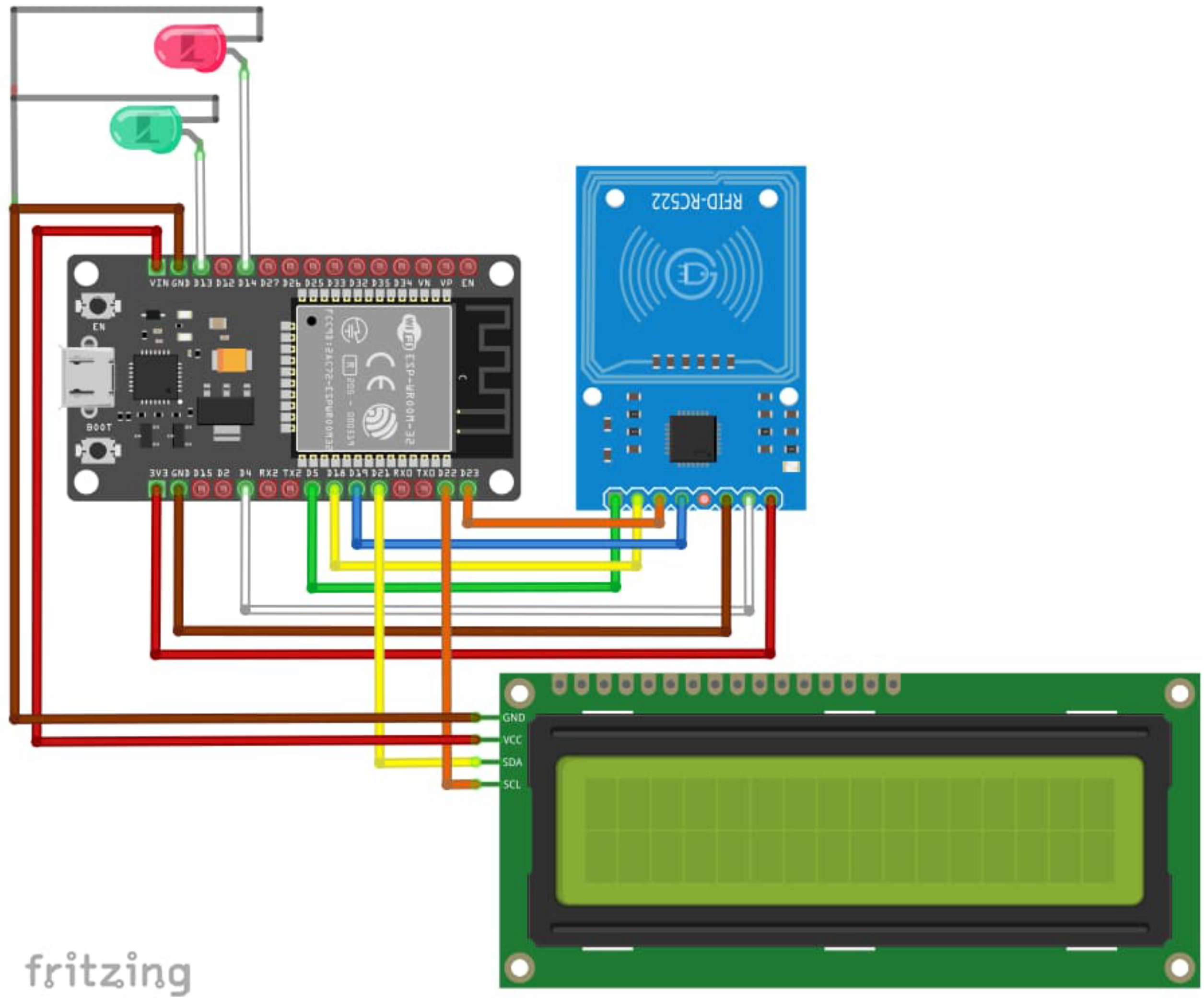
Safe GPIOs for SPI and peripherals include 5, 18, 19, 23, 27, 32, 33[10] [11]. Avoid strapping pins (0, 2, 12, 15) unless boot modes are understood[1].

3.2 Connection Table

RFID RC522	ESP32 GPIO	Function
VCC	3V3	+3.3 V supply
GND	GND	common ground
RST	27	reset control
MISO	19	SPI MISO
MOSI	23	SPI MOSI
SCK	18	SPI CLK
SDA/SS	5	SPI CS

Buzzer signal attaches to GPIO 15 with the other lead to GND for active modules[12] [13].

The complete layout is illustrated below.


RFID-RC522
VIN GND D13 D12 D14 D27 D26 D25 D33 D32 D35 D34 VN VP EN
EN
BOOT
ESP-WROOM-32
3V3 GND D15 D2 D4 RX2 TX2 D5 D18 D19 D21 RX0 TX0 D22 D23
GND
VCC
SDA
SCL
fritzing

Fritzing wiring diagram showing an ESP32 connected to an RFID-RC522 module, an I2C LCD, and two LEDs.

4 — Firmware Architecture

4.1 Library Stack

- WiFi.h → network join[5].
- HTTPClient.h → POST JSON payloads[5].
- MFRC522.h → UID reading & authentication[14].
- Preferences.h → non-volatile daily limit storage[8] [15].
- ArduinoJson.h → compact serialization[2].

4.2 Core Workflow (Pseudo-Code)

```
loop():
  if rfid card present:
      beep_short()
      uid = read_uid()
      activity = map_uid(uid)
      if activity == "" → reject()
      else if used_today(uid) → reject()
      else:
          log_to_sheet(uid, activity, "success")
          record_last_use(uid)
          triple_beep()
```

4.3 Google Apps Script

```
function doPost(e){
  const sheet = SpreadsheetApp.getActiveSheet();
  const data = JSON.parse(e.postData.contents);
  sheet.appendRow([new Date(), data.uid, data.activity, data.status]);
  return ContentService.createTextOutput("OK");
}
```

Deploy as *Anyone* but add an optional secret key in the JSON to mitigate spam[2] [6].

4.4 Daily Restriction Logic

```
bool used_today(String uid){
  prefs.begin("cards", false);
  String last = prefs.getString(uid, "");
  String today = String( (uint32_t)(millis()/86400000) );   // UTC days since boot
  if(last == today){
     prefs.end();
     return true;
  }
  prefs.putString(uid, today);
  prefs.end();
  return false;
}
```

Flash endurance is preserved because each UID writes once per day, well below 10 k cycles[16].

5 — Power Management

Active Wi-Fi transmissions can draw 285 mA peaks[17]. Average current during logging (~80 mA) can be slashed to <200 µA by entering deep-sleep between scans and waking on GPIO interrupt from the RFID reader's IRQ (version 2 feature)[3] [18] [19]. For battery-backed builds use a TP4056 + boost converter to 5 V per standard IoT power packs[20] [21].

6 — Assembly & Prototyping Tips

- Seat the ESP32 straddling the breadboard valley to expose two rows per pin for jumpers[22].
- Keep SPI wires under 10 cm to avoid signal integrity issues at 4 MHz[23] [24].
- Label jumper bundles with masking tape by function (SCK, MOSI, etc.) for clarity[25].

- Test Wi-Fi credentials and Script URL via serial debug before mounting in an enclosure[26] [27].

7 — Testing Procedure

1. Flash *ReadNUID* example to verify UID detection[14].
2. Update `registeredCards[][^28]` with printed UIDs and desired activity tags.
3. Flash main sketch; open Serial Monitor at 115200 baud.
4. Tap a valid card → expect single short beep, then three quick beeps after "200 OK" POST.
5. Re-tap same card within the day → expect triple long reject beeps.

8 — Enclosure & Ergonomics

A 52 × 57 × 25 mm PLA case with cutouts for USB-C and antenna keeps the stack portable; STL files for generic DevKit boards are freely downloadable[28] [27]. Mount the RFID antenna flush to the lid for optimal coupling (2–3 cm read range)[29].

9 — Future Expansion (Roadmap)

Version	Added Features	Key Components
1.1	RGB LED status	WS2812-B
1.2	OLED habit dashboard	SSD1306 I^{2}C
2.0	Battery + deep-sleep wake	TP4056, 18650
2.5	BLE phone sync	ESP-NOW/BLE
3.0	Local SD logging fail-safe	MicroSD module

Conclusion

The documented design demonstrates that an ESP32-based RFID logger, with minimal parts and Apps Script integration, offers a low-friction way to quantify daily routines while enforcing one-tap-per-day goals. Sound cues and optional LEDs provide immediate feedback, and future firmware can leverage deep-sleep to extend battery autonomy for field use. Adhering to the wiring pinout, flash-safe Preferences usage, and secure HTTPS endpoints will ensure long-term reliability and data integrity[1] [4] [30].

⁂

1. https://www.electronifyindia.com/blogs/news/esp32-wroom-32-datasheet
2. https://www.oceanlabz.in/sending-data-from-esp32-to-google-sheets-iot-data-logger-tutorial/
3. https://www.programmingelectronics.com/esp32-deep-sleep-mode/
4. https://randomnerdtutorials.com/esp32-datalogging-google-sheets/
5. https://how2electronics.com/rfid-rc522-attendance-system-using-arduino/
6. https://electropeak.com/learn/sending-data-from-esp32-or-esp8266-to-google-sheets-2-methods/