# ARDUINO SIX-PARTY EVM PROTOTYPE

Detailed project documentation for an educational button-operated voting-machine demonstration

| | |
|---|---|
| PLATFORM | **Arduino Uno R3** |
| INTERFACE | **128x64 I2C OLED + Serial Monitor** |
| INPUTS | **6 party buttons + green confirm + red cancel** |
| STORAGE | **Persistent EEPROM vote totals** |
| CREDIT | **made by Chitraksh AND Ansh did not make it** |

Version 1.0 | 18 August 2026

# 1. Project overview

The Arduino Six-Party EVM Prototype is a compact embedded-systems project that demonstrates the interaction flow of an electronic voting machine. A voter selects one of six fictional parties, checks the full party name, and either confirms with a green button or cancels with a red button. Confirmed votes are counted and stored in non-volatile EEPROM.

| | |
|---|---|
| **Purpose** | Demonstrate physical input handling, confirmation logic, persistent data, protected operator access, and a small-display user interface on an Arduino Uno. |

## Key capabilities

- Six dedicated party-selection buttons with full-name confirmation.
- Separate green confirm and red cancel buttons.
- 30 ms software debouncing and INPUT_PULLUP wiring.
- Vote totals retained in EEPROM after power loss or reset.
- Protected results screen with leader and tie reporting.
- A deliberate three-second hold gesture for clearing all stored votes.
- Complete Serial Monitor interface for testing without the OLED.
- Post-vote thank-you and exact creator-credit screen.

## Scope and limitations

This is an educational demonstration, not a real election system. It has no voter authentication, cryptographic ballot protection, certified audit trail, secure boot, tamper detection, independent verification, accessibility certification, or election-authority approval.

| | |
|---|---|
| **Safety statement** | Do not use this prototype to conduct a public, institutional, or legally consequential election. |

# 2. Functional specification

## Voter requirements

- The ready screen invites the voter to press one party button.
- After selection, the interface displays both the party number and full party name.
- GREEN records exactly one vote; RED cancels without changing totals.
- A saved vote displays `Thank you`, followed by the exact credit text.
- The machine returns automatically to the ready state for the next voter.

## Configured fictional parties

| Button | Party name | Arduino input |
|---|---|---|
| 1 | NATKHAT PARTY | D2 |
| 2 | AURORAEON JANTA PARTY | D3 |
| 3 | MACHHAR PARTY | D4 |
| 4 | AVENGERS PARTY | D5 |
| 5 | MADHUMAKHI PARTY | D6 |
| 6 | BUCCHU PARTY | A0 |

## Operator requirements

- Hold GREEN and RED together for two seconds from VOTING READY to open passcode entry.
- Use party buttons as digits 1 through 6; GREEN submits and RED deletes or exits.
- The configured demonstration passcode is 2614.
- Results show all six totals and every party tied for the highest total.
- From RESULTS, hold GREEN for three seconds to reset all totals to zero.

**Passcode note** The hard-coded passcode is suitable only for demonstration. It must not be treated as security for a real election.

# 3. Hardware and bill of materials

## Primary hardware

| Item | Qty | Purpose / specification |
|---|---|---|
| Arduino Uno R3 | 1 | ATmega328P controller running the voting logic |
| 0.96-inch OLED | 1 | 128x64, four-pin I2C, SSD1306-compatible, address 0x3C |
| Party push buttons | 6 | Dedicated inputs for parties 1 through 6 |
| Green push button | 1 | Confirm, submit passcode, hold to reset results |
| Red push button | 1 | Cancel, delete passcode digit, return from results |
| Breadboard / terminals | 1 | Prototype interconnection and shared ground |
| Jumper wires | As needed | Signal, 5 V, and ground connections |
| USB data cable | 1 | Programming, 5 V power, and Serial Monitor |

## Electrical approach

Each push button connects between its assigned Arduino pin and GND. The firmware configures every input as INPUT_PULLUP, so an unpressed button reads HIGH and a pressed button reads LOW. No external pull-up resistors are required for the buttons.

The four-pin OLED uses the Uno's I2C bus. On modules labeled SCK rather than SCL, SCK is the I2C clock connection for this board.

**Power caution** Always remove USB power before changing wires. Connect by printed pin labels, not by assuming the physical header order.

# 4. Exact wiring

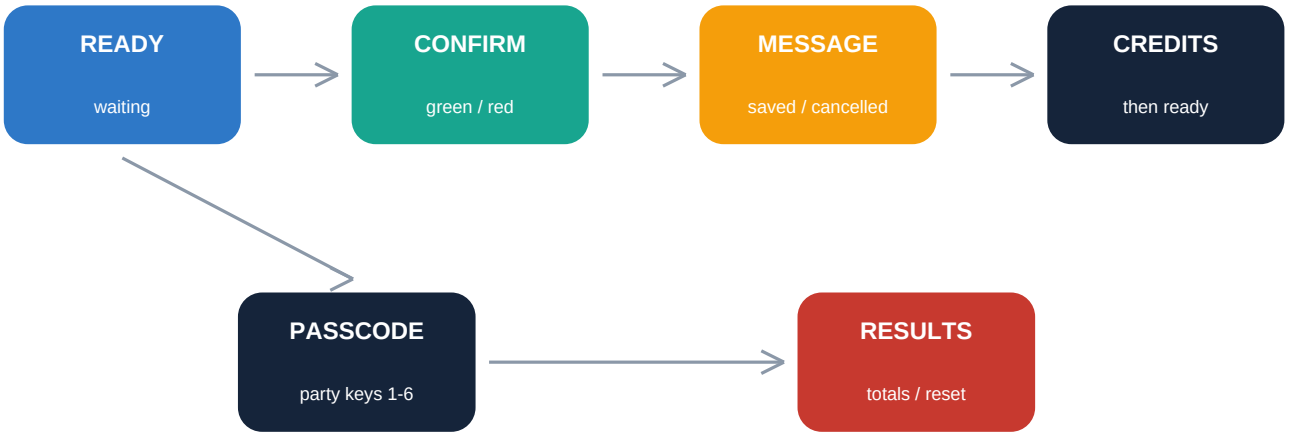
| Device / control | Module pin | Arduino Uno | Connection notes |
|---|---|---|---|
| OLED | GND | GND | Common ground |
| OLED | VCC | 5V | For the identified 3-5 V-compatible module |
| OLED | SCK | A5 | I2C clock; equivalent to SCL on this module |
| OLED | SDA | A4 | I2C data |
| Party 1 | Button | D2 | Other terminal to GND |
| Party 2 | Button | D3 | Other terminal to GND |
| Party 3 | Button | D4 | Other terminal to GND |
| Party 4 | Button | D5 | Other terminal to GND |
| Party 5 | Button | D6 | Other terminal to GND |
| Party 6 | Button | A0 | Used as a digital input; other terminal to GND |
| GREEN | Button | A1 | Used as a digital input; other terminal to GND |
| RED | Button | A2 | Used as a digital input; other terminal to GND |

## Connection sequence

- Disconnect USB power.
- Create a shared GND rail and connect every button's common terminal.
- Connect party and control signal pins exactly as listed.
- Connect OLED GND and VCC, then SCK to A5 and SDA to A4.
- Inspect for loose strands, adjacent-pin bridges, and reversed VCC/GND.
- Reconnect USB and verify the Arduino ON indicator remains steady.

# 5. Firmware architecture

The sketch uses a small explicit state machine rather than long blocking delays. This keeps button scanning responsive while controlling what each button means on each screen.




Operator branch: hold GREEN + RED, authenticate, inspect results

## State responsibilities

| State | Responsibility | Important transitions |
|---|---|---|
| READY | Listens for party selection and the operator hold gesture | Party -> CONFIRM; G+R hold -> PASSCODE |
| CONFIRM | Shows number and full name | GREEN -> MESSAGE; RED -> MESSAGE |
| MESSAGE | Shows saved, cancelled, wrong-code, or reset feedback | Timer -> CREDITS or READY |
| CREDITS | Displays exact creator text for three seconds | Timer -> READY |
| PASSCODE | Collects four digits using party buttons | Correct+GREEN -> RESULTS |
| RESULTS | Shows totals, leaders, return, and reset controls | RED -> READY; GREEN hold -> reset |

## Timing constants

| Constant | Value | Purpose |
|---|---|---|
| DEBOUNCE_MS | 30 ms | Reject mechanical contact bounce |
| MESSAGE_MS | 1600 ms | Duration of feedback messages |
| CREDIT_MS | 3000 ms | Duration of creator-credit screen |
| ADMIN_HOLD_MS | 2000 ms | Intentional operator-entry hold |
| RESET_HOLD_MS | 3000 ms | Intentional destructive reset hold |
| OLED_KEEP_AWAKE_MS | 5000 ms | Periodic display-on and framebuffer refresh |

# 6. Data storage and interfaces

## EEPROM persistence

Six 32-bit unsigned counters store party totals. The sketch first checks a 16-bit magic value (0x564D). If the marker is missing, the counters are initialized; otherwise, stored totals are loaded. Confirmed votes and explicit resets are written back to EEPROM.

| EEPROM region | Contents | Size |
|---|---|---|
| Address 0 | Magic marker 0x564D | 2 bytes |
| Address 2 onward | Six uint32_t vote totals | 24 bytes |

**Write-cycle note** EEPROM has finite write endurance. This demonstration writes only when a vote is confirmed or all totals are reset; it does not write continuously.

## OLED interface

The code targets a 128x64 SSD1306-compatible OLED at I2C address 0x3C using Adafruit SSD1306 and Adafruit GFX. `USE_OLED` is currently set to 1. If initialization fails, the sketch continues operating through Serial Monitor.

## Serial Monitor interface

At 9600 baud, Serial Monitor reproduces the full user interface: ready instructions, every debounced button event, selected party name, confirmation instructions, masked passcode progress, result totals, leaders, reset confirmation, and the exact creator credit.

## Representative output

```
BUTTON: PARTY 4 - AVENGERS PARTY
CONFIRM VOTE
Selected: Party 4 - AVENGERS PARTY
BUTTON: GREEN
VOTE SAVED
Thank you
made by Chitraksh AND Ansh did not make it
```

# 7. Operating procedure

## Voter workflow

| Step | Action | Expected response |
|---|---|---|
| 1 | Wait for VOTING READY | Party-button instructions are visible |
| 2 | Press one party button | Number and full party name appear |
| 3 | Check the displayed name | GREEN and RED options are shown |
| 4 | Press GREEN to confirm | Vote increments and VOTE SAVED appears |
| 5 | Or press RED to cancel | No total changes and VOTE CANCELLED appears |
| 6 | After a saved vote | Thank you, exact credit text, then READY |

## Operator workflow

| Step | Action | Expected response |
|---|---|---|
| 1 | At READY, hold GREEN+RED for 2 seconds | ADMIN PASSCODE appears |
| 2 | Release both buttons | No held input interferes with entry |
| 3 | Press Party 2, 6, 1, 4 | Masked field advances to **** |
| 4 | Press GREEN once | RESULTS lists all totals and leaders |
| 5 | Press RED | Returns to VOTING READY |
| Reset | In RESULTS, hold GREEN for 3 seconds | All totals become zero and persist |

**Reset is destructive** Only reset after results have been recorded or when beginning a new demonstration. The operation overwrites all six stored totals with zero.

# 8. Software setup and upload

## Required software

- Arduino IDE with the Arduino AVR Boards package.
- Adafruit SSD1306 library.
- Adafruit GFX Library dependency.
- Wire and EEPROM, which are included with the Arduino platform.

## Upload procedure

| Step | Procedure |
|---|---|
| 1 | Open `OLED_Voting_Machine.ino` in Arduino IDE. |
| 2 | Install Adafruit SSD1306 and Adafruit GFX through Library Manager. |
| 3 | Select Tools > Board > Arduino Uno. |
| 4 | Select the detected serial port. |
| 5 | Click Verify, then Upload. The Uno normally resets automatically. |
| 6 | Open Serial Monitor and select 9600 baud. |
| 7 | Confirm VOTING READY and test all eight buttons before use. |

## Important configuration

```
#define USE_OLED 1
constexpr uint8_t OLED_ADDRESS = 0x3C;
const char ADMIN_PASSCODE[] = "2614";
```

For display-free testing, set `USE_OLED` to 0. In that mode, the OLED headers and framebuffer are excluded and Serial Monitor remains the complete interface.

# 9. Verification and troubleshooting

## Recommended test checklist

| Test | Method | Pass condition |
|---|---|---|
| Startup | Power the Uno and open Serial Monitor | One startup banner and VOTING READY |
| Party inputs | Press Party 1 through 6 | Correct number and full name for each |
| Controls | Press GREEN and RED in confirmation | Save and cancel paths behave correctly |
| Persistence | Record a vote, restart, open results | Total remains stored |
| Authentication | Try a wrong code, then 2614 | Wrong denied; correct code opens results |
| Tie logic | Create equal leading totals | Every tied leader is reported |
| Reset | Hold GREEN in results for 3 seconds | All totals become and remain zero |
| Credit | Confirm a vote | Exact sentence appears after Thank you |

## Known prototype issue

**OLED intermittently blanks**

On the physical prototype, the generic OLED has reportedly turned black after several seconds. The firmware currently reasserts normal mode, DISPLAY ON, and the framebuffer every five seconds, but this has not yet eliminated the symptom. The issue remains under investigation and must not be represented as solved.

## Controlled diagnosis

- Observe whether the Arduino ON indicator remains steady when the OLED blanks.
- Watch Serial Monitor for a repeated startup banner, which would prove a reset or brownout.
- Run an I2C scanner and record whether the module continues acknowledging 0x3C after blanking.
- Measure 5 V and GND at the OLED during normal and blank states.
- Test the standard Adafruit 128x64 I2C example to separate hardware behavior from application logic.
- Verify whether the delivered controller is truly SSD1306-compatible or requires an SH1106 driver.

# 10. Engineering decisions and future work

## Notable design decisions

- Analog pins A0-A2 are reused as digital button inputs to conserve Uno pins.
- I2C uses only A4 and A5, minimizing OLED wiring and avoiding the earlier SPI display complexity.
- All human-facing selections include full party names, reducing ambiguity.
- Reset requires a long hold to reduce accidental data destruction.
- Serial output mirrors the UI, making hardware-independent testing possible.
- Flash-memory strings reduce SRAM pressure on the ATmega328P.

## Future improvements

| Priority | Improvement | Benefit |
|---|---|---|
| High | Resolve OLED blanking with measured power/I2C evidence | Stable standalone presentation |
| High | Add a sealed operator reset procedure and result export | Better control and auditability |
| Medium | Store redundant counters with checksums | Detect EEPROM corruption |
| Medium | Add enclosure, labels, strain relief, and keyed connectors | Safer and more reliable hardware |
| Medium | Add a buzzer or status LED with accessibility review | Clearer interaction feedback |
| Low | Create automated input simulation tests | Repeatable regression testing |

## Portfolio summary

This project demonstrates practical embedded development: translating an interaction idea into pin-level hardware, implementing a non-blocking state machine, handling noisy physical inputs, persisting data, building separate voter and operator flows, and iterating through display and voltage-compatibility problems.

| | |
|---|---|
| **Suggested portfolio tags** | Arduino \| Embedded C++ \| Electronics \| I2C \| OLED \| EEPROM \| Human-machine interface |

# Appendix A. Project assets

| Asset | Purpose |
|---|---|
| OLED_Voting_Machine.ino | Current Arduino firmware |
| arduino_evm_project_documentation.pdf | Technical documentation for portfolio download |
| evm_portfolio_website_prompt.md | Implementation prompt for editing the portfolio site |
| voter_guide_A5_4page.pdf | Four-page voter reference |
| operator_guide_A5_4page.pdf | Four-page operator reference |
| voter_booklet_A4_duplex_print.pdf | Foldable A4 duplex voter booklet |
| operator_booklet_A4_duplex_print.pdf | Foldable A4 duplex operator booklet |
| fictional_party_strip_A4_print.pdf | Printable party-number and symbol strip |

## Appendix B. Exact public-facing project copy

| | |
|---|---|
| **Title** | Arduino Six-Party EVM Prototype |
| **Short description** | A button-operated educational voting-machine prototype with confirmation controls, persistent vote storage, protected result viewing, and OLED/Serial feedback. |
| **Credit** | made by Chitraksh AND Ansh did not make it |

## Document control

| Field | Value |
|---|---|
| Document version | 1.0 |
| Date | 18 August 2026 |
| Firmware source reviewed | OLED_Voting_Machine.ino |
| OLED mode | Enabled (`USE_OLED 1`) |
| Known issue status | OLED intermittent blanking under investigation |

End of project documentation