NoRisk

Bluetooth Automated Smart Access

RISC-V Ethical IOT Hackathon

15 June 2024

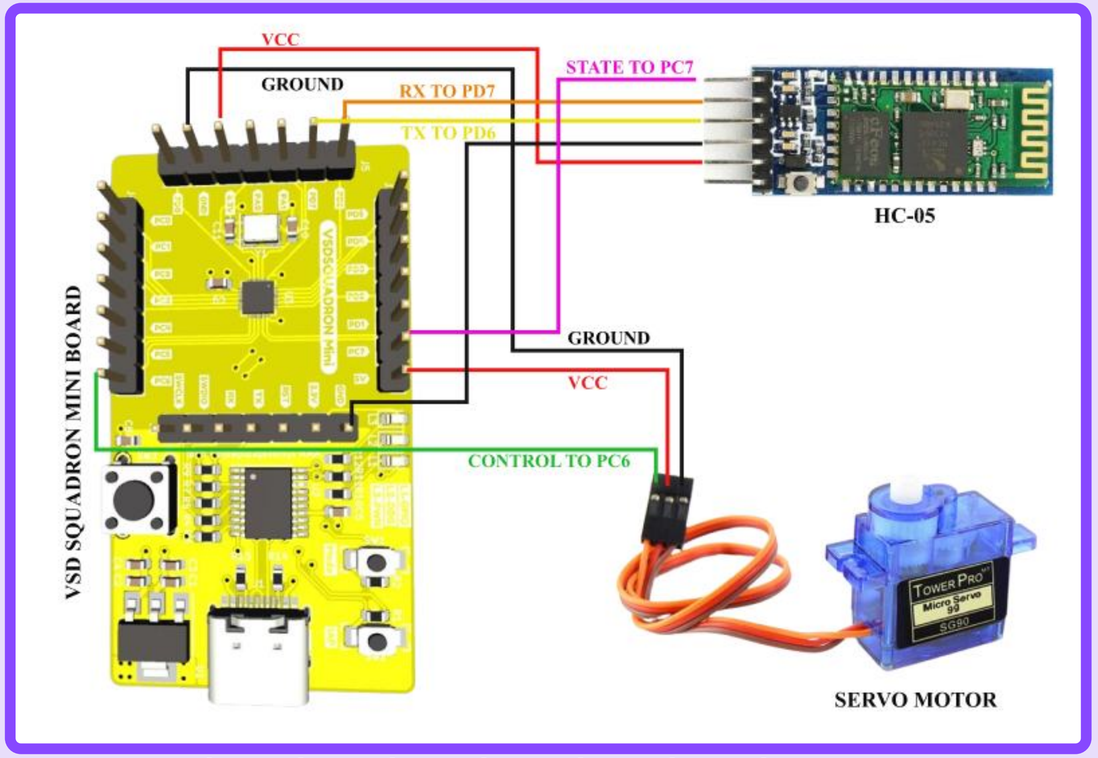
International Institute of Info Tech, Bangalore

Introduction

In an age of seamless technology integration, smart homes have become a tangible reality. Central to this transformation is the "***Bluetooth Automated Smart Access with Two Factor Authentication***" system, blending convenience, security, and innovation.

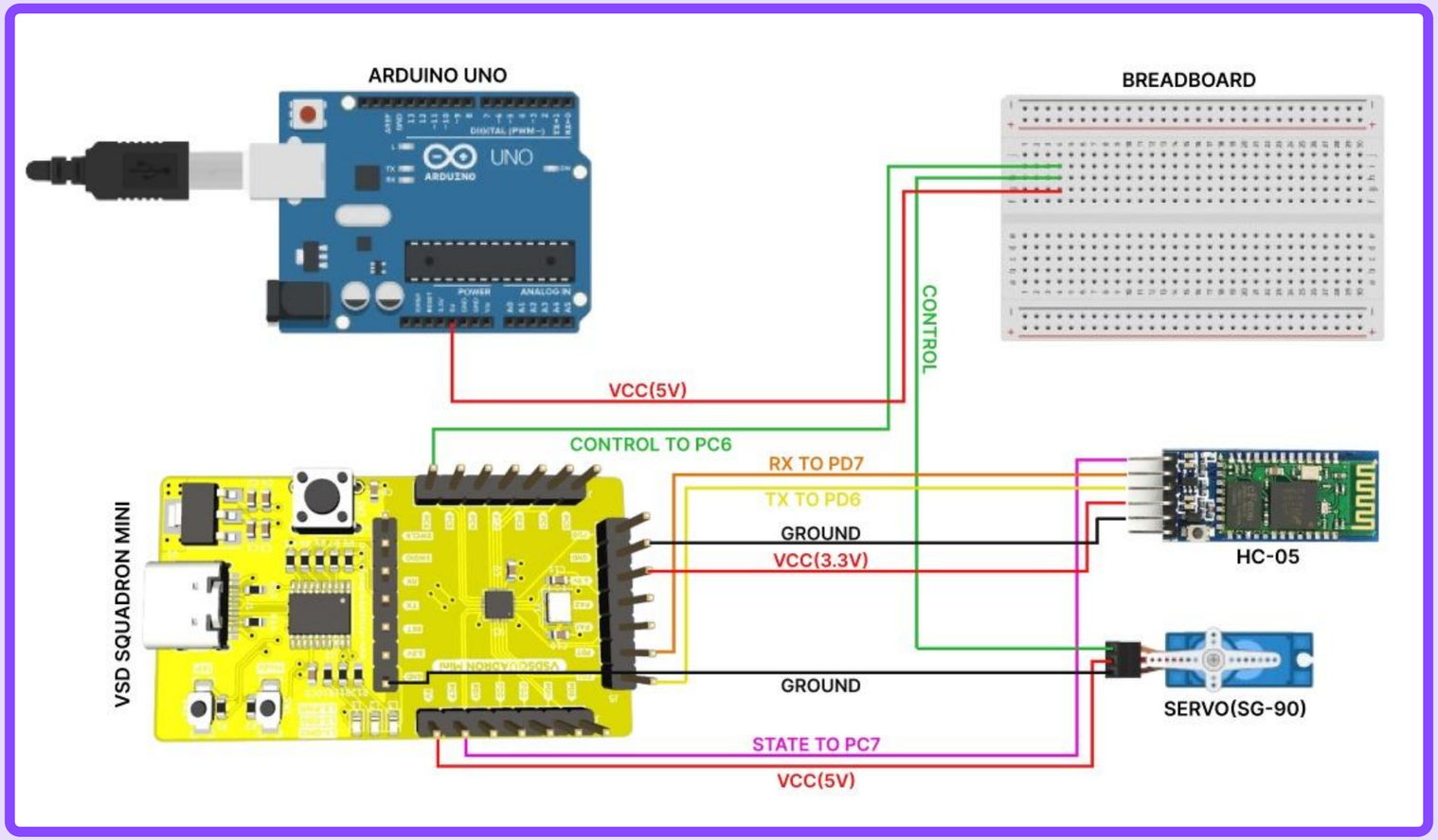
Utilising a ***VSD Squadron Mini board***, a ***Bluetooth module***, and a ***servo motor***, this system enhances interaction with living spaces. It replaces cumbersome manual operations with smartphone management, improving efficiency.

Applications range from controlling water taps and lighting to smart door security, simplifying everyday tasks across various environments.

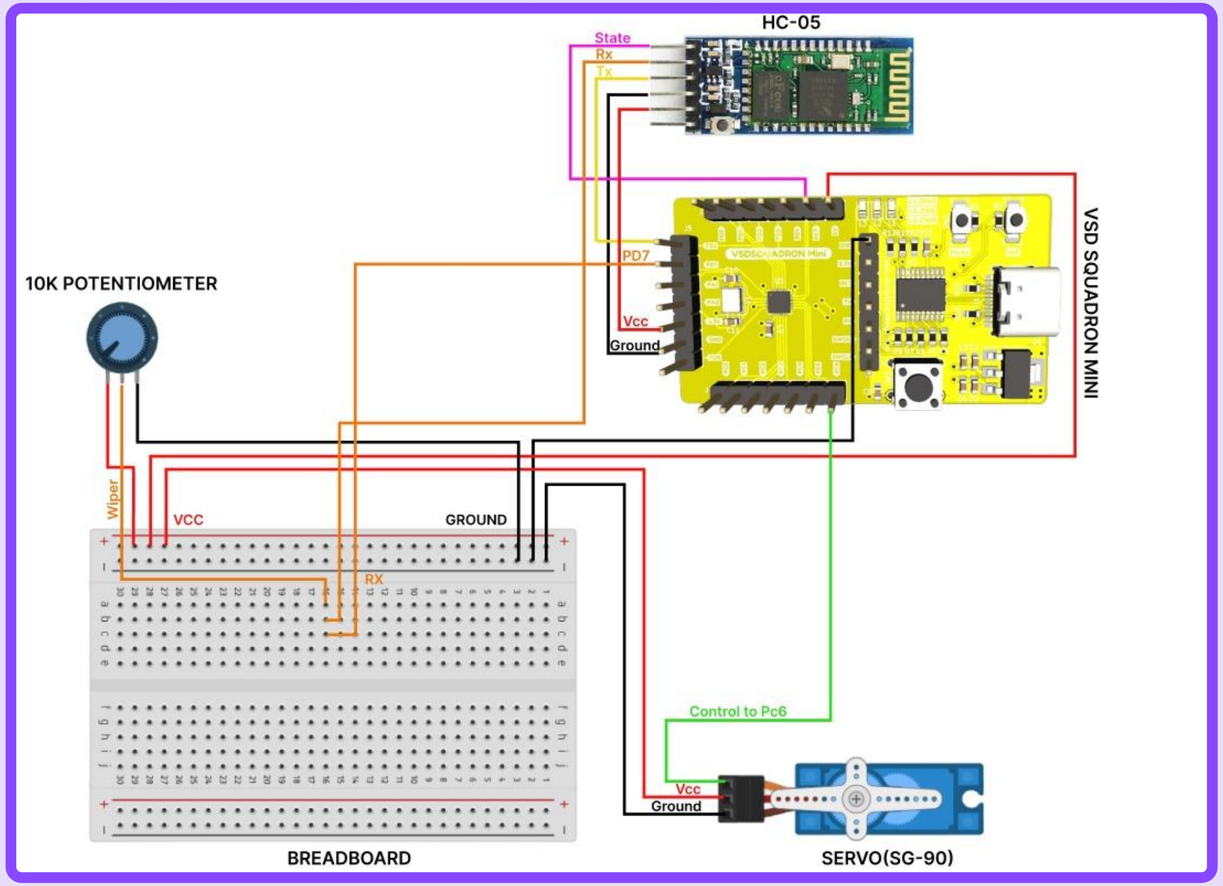


The Underlying Principle

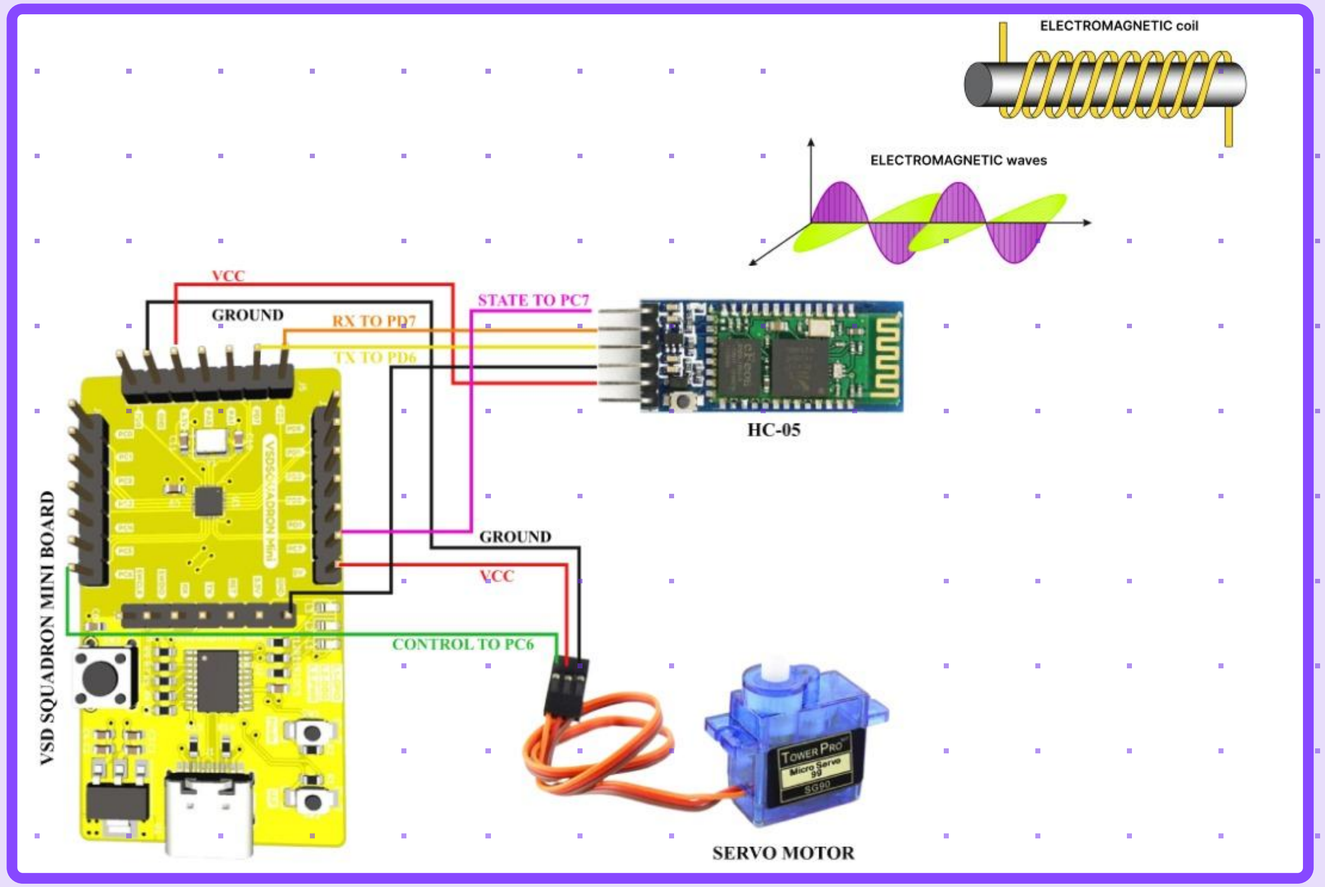
Hardware Fault Injection 1:
Disrupting Servo Motor using external Arduino 5V Power Source



Hardware Fault Injection 2:
Inducing fault in HC-05 using potentiometer to corrupt receiver signal (RX)



Hardware Fault Injection 3:
Disrupting bluetooth signal using EM Waves



Software Fault Injection 1: Simulate - faultCondition

The fault condition is triggered by any command between 'A' and 'Z' (excluding 'O' and 'C') within checkBluetooth() function. This sets the faultCondition flag to true, which locks the door and skips the rest of the loop.

```
void checkBluetooth() {
  if (Serial.available()) {
    char command = Serial.read();
    // other open and close codes..
    if ((command >= 'A' && command <=
'Z') && command != 'O' && command !=
'C') {
      faultCondition = true;
      Serial.println("Fault simulation
command received");
    }
  }
}
```

```
void loop() {
  // Check for fault condition
  if (faultCondition) {
    Serial.println("Fault Detected!
System is locking the door.");
    lockDoor(); // Lock the door
immediately
    return; // Skip the rest of the
loop
  }
  // Other code...
}
```

Software Fault Injection 2: Simulate - rebootSystem

rebootSystem() simulates a system reboot by resetting specific variables and calling setup() to reinitialise the system which is randomly triggered based on a timer within the loop() function.
Every second (rebootCheckInterval), the system checks if a random event should trigger a reboot with a 50% chance.

```
void loop() {
  // Check if it's time to randomly
reboot
  if (millis() - lastRebootCheck >
rebootCheckInterval) {
    lastRebootCheck = millis();
    if (random(0, 100) < 50) { // 50%
chance to reboot
      Serial.println("Random reboot
triggered!");
      rebootSystem();
    }
  }
  // Other code...
}
```

```
void rebootSystem() {
  // Simulate a system reboot by
resetting variables and re-
initializing
  Serial.println("Rebooting
system...");
  doorLocked = true;
  faultCondition = false;
  lastRebootCheck = millis();
  setup(); // Call setup to re-
initialize the system
}
```

Hardware Fault Protection

1. Arduino Fault Protection

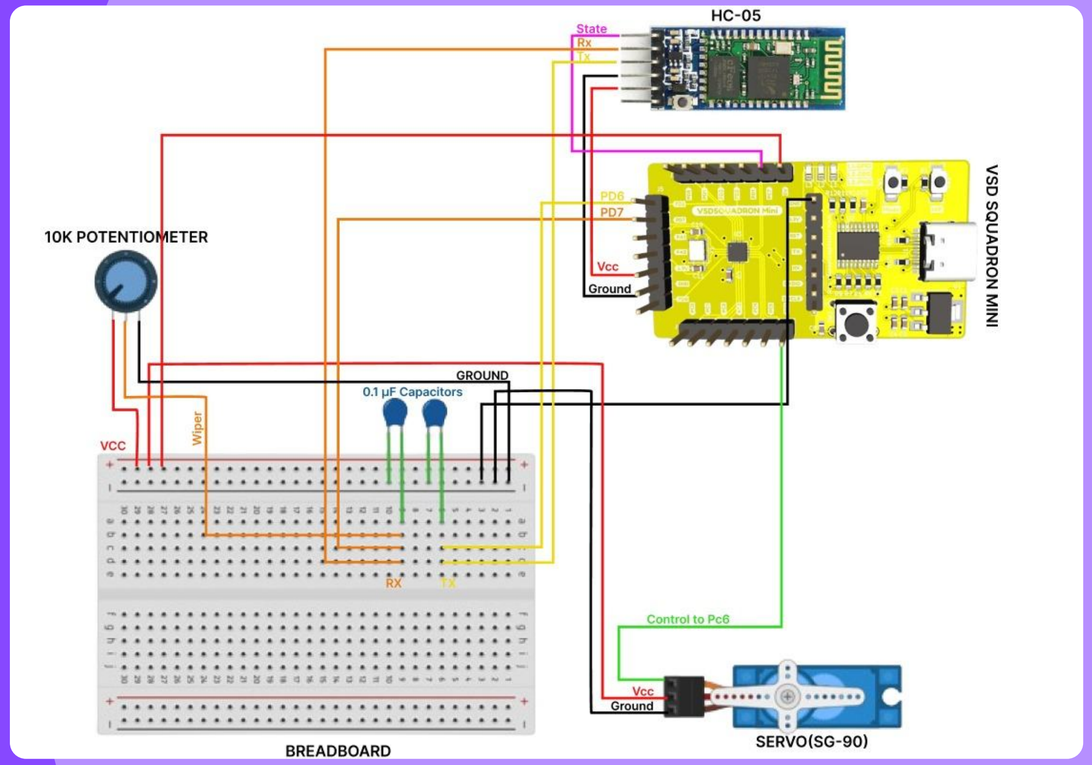
This fault can be protected by ensuring a safe enclosure where access is denied to manipulate the input voltage levels to the servo motor.

2. Noise Signal Fault Protection

The capacitors added in this circuit act as a low pass filter which smoothen out any irregularities in the voltage fluctuations

3. EM Interference Protection

A Faraday cage, by enclosing the components in a conductive material, can be employed to shield sensitive Bluetooth components.



Circuit Representing Noise Signal Fault Protection

Software Fault Protection

1. Debounce Protection: Ensures a minimum interval between commands to prevent spamming.
2. Authorisation Checks: Verifies that the command matches the authorised code before executing further actions.

```
bool debounceProtection() {
  unsigned long currentTime =
millis();
  if (currentTime - lastCommandTime >
commandInterval) {
    lastCommandTime = currentTime;
    return true;
  } else {
    Serial.println("Command ignored
due to debounce protection.");
    return false;
  }
}
```

```
bool verifyAuthorization(String
command) {
  return command == authorizedCode;
}
```

Software Fault Protection

3. Scheduled Reboots: Reboots the system at regular intervals to prevent long-term instability.
4. Reboot System: Resets critical variables and simulates a system reboot at predetermined intervals based on the loopCounter.

```
void loop() {
  // Increment loop counter
  loopCounter++;

  // Other code...

  // Check if it's time to reboot
based on loop counter
  if (loopCounter >= rebootThreshold)
{
    Serial.println("Rebooting
system...");
    rebootSystem();
  }
}
```

```
void rebootSystem() {
  doorLocked = true;
  faultCondition = false;
  loopCounter = 0;
  lastRebootTime = millis();
  setup();
}
```

Applications

Railways

Water Dispenser

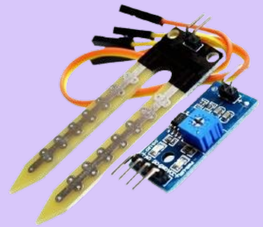
Zoo Catering

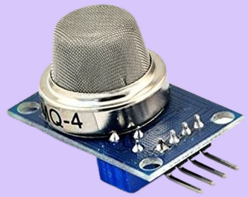
Locker

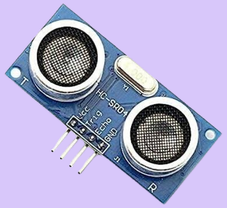
Smart Door

Lawn Sprinkler

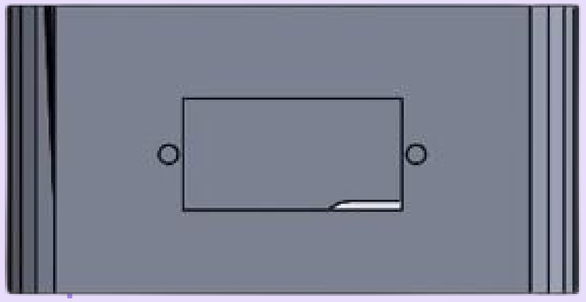
Sensor Integration

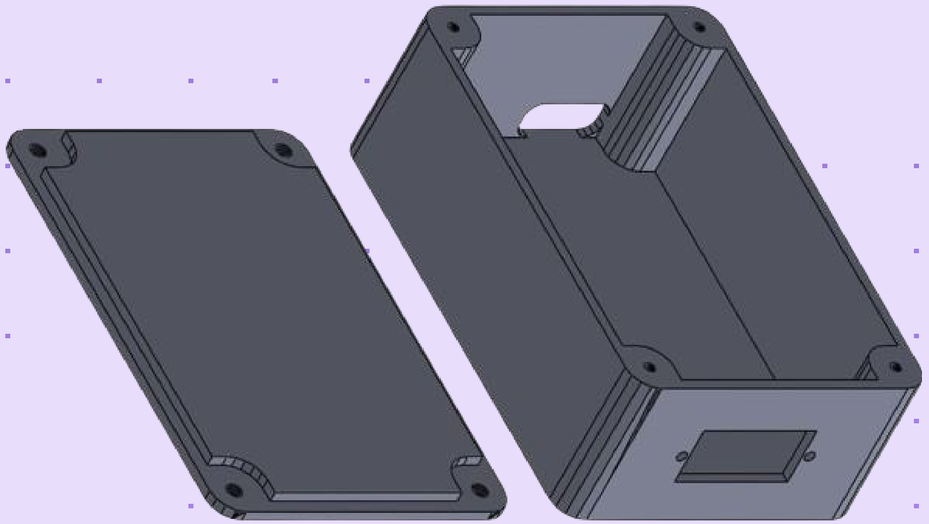






3D Printed Enclosure Design





- COMPACT
- FUNCTIONAL
- RELIABLE
- SECURE

Thanks for your time !

NoRisk - a Project by *Aishwarya* & *Mahathi*

https://github.com/rmahathi/NoRisk