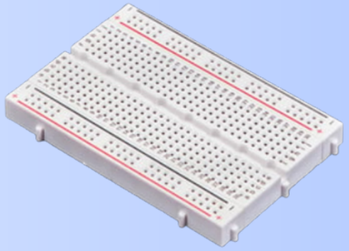
Università
degli Studi di
Trento

SMART HOME

COMPONENTS

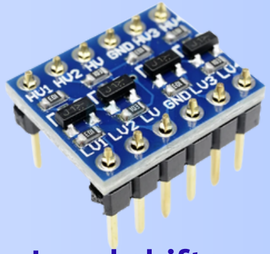
MSP432 Launchpad



Breadboard

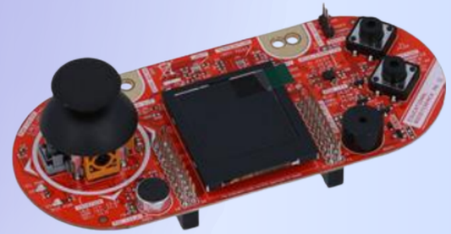
Arduino Uno

Microservi SG90

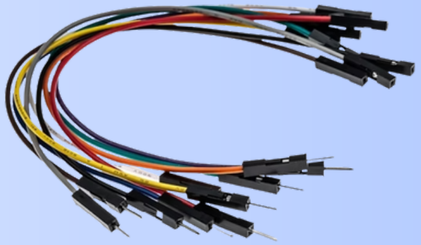


Level shifter

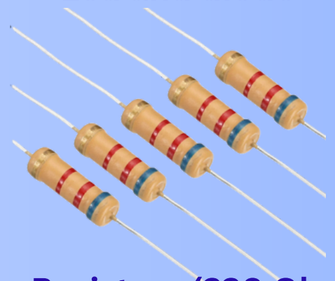
Led and diode



MSP432 Boosterpack Sensors



Jumpers



Resistors (220 Ohm)

PROJECT OVERVIEW & PROBLEM STATEMENT

The goal of this project is to develop a smart home automation system using microcontrollers for the management automation of home functionalities:

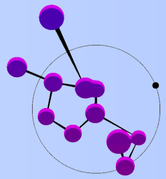
- lighting
- temperature
- earthquake detection

The MSP432 processes sensor data and makes decisions, while the Arduino Uno handles communication and actuation. The most challenging part is making the devices communicate reliably, with real-time response.

SYSTEM ARCHITECTURE & DATA FLOW

```
switch (idx) {
  case 0: Serial1.write('O'); break;  // Turn ON lights
  case 1: Serial1.write('F'); break;  // Turn OFF lights
  case 2: Serial1.write('D'); break;  // Open window
  case 3: Serial1.write('H'); break;  // Close window
  case 4:                             // Enter Smart House
    Serial1.write('I');               //  Open door
    (...)
  case 5:                             // Exit Smart House
    Serial1.write('I');
```

Master code



- Master Code (MSP432) - manages the control logic of the smart home system. It reads data from sensors (temperature, light, accelerometer), detects earthquakes, and provides a user-friendly menu for switching between AUTO and EDIT modes. Commands are sent to the Arduino via UART.

- Slave Code (Arduino) - receives commands from the MSP432 and directly controls the hardware components (servos for door/window, LEDs for lighting and alerts). Executes low-level actions based on the master's instructions.

```
switch (command) {
  case 'H': windowServo.write(0); break;     // Close window
  case 'D': windowServo.write(180); break;   // Open window
  case 'N': windowServo.write(90); break;    // Neutral window

  case 'I': doorServo.write(180); break;     // Open door
  case 'U': doorServo.write(0); break;       // Close door

  case 'O': digitalWrite(whiteLightPin, HIGH); break;  // Lights ON
  case 'F': digitalWrite(whiteLightPin, LOW); break;   // Lights OFF

  case 'T':                                  // Earthquake alert
    earthquakeOngoing = true;
    earthquakeStartTime = millis();
    blinkTimer = millis();
    break;
}
```

Slave code

CODE STRUCTURE & CONTROL LOGIC

State machine & mode handling

```
// Menu and state control
enum State { MENU, AUTO, EDIT };
```

- System runs as a finite state machine with 3 modes: MENU, AUTO, EDIT
- AUTO mode reacts to buttons and sensors every 4 seconds using millis()
- Door logic uses a 5-second timer to handle automatic closing

Entry/Exit logic

```
// ENTRY (INSIDE): open door and turn on lights
if (!doorMoving && digitalRead(buttonOne) == LOW && !inside) {
  doorMoving = true;
  doorTimer = now;
  Serial1.write('I'); delay(50); Serial1.write('O');
}

// EXIT (OUTSIDE): open door and turn off lights
if (!doorMoving && digitalRead(buttonTwo) == LOW && inside) {
  doorMoving = true;
  doorTimer = now;
  Serial1.write('I'); delay(50); Serial1.write('F');
}

// After 5 seconds, close door
if (doorMoving && now - doorTimer >= 5000) {
  Serial1.write('U');
  inside = !inside;
  doorMoving = false;
}
```

- On entry: sends 'I' to open door, then 'O' to turn on lights
- On exit: sends 'I', then 'F' to turn off lights
- After 5s → sends 'U' to close door

Earthquake handling

- Continuously checks for vibration across 3 axes
- If shake is detected:
- Activates buzzer + red LED
- Sends 'T' to Arduino
- After 5s of stability → alert stops automatically

```
// Check if motion exceeds threshold
bool shake = abs(xVal - calibX) > 120 || abs(yVal - calibY) > 120 || abs(zVal - calibZ) > 120;

//Activate earthquake alert
if (shake && !earthquake) {
  earthquake = true;
  tone(buzzerPin, 1000);              // Play buzzer
  digitalWrite(redLED, HIGH);         // Turn on red LED (MSP432)
  Serial1.write('T');                 // Alert Arduino (turn on red LED)
  earthquakeEndTimer = millis();
  displayNeedsUpdate = true;
}

// Stop alert after 5s of no shake
if (!shake && earthquake) {
  if (millis() - earthquakeEndTimer >= 5000) {
    earthquake = false;
    noTone(buzzerPin);                // deactivate buzzer
    digitalWrite(redLED, LOW);        // Turn off red LED (MSP432)
    displayNeedsUpdate = true;
  }
}
```

TESTING, ISSUES AND DEBUGGING

```
void simulateCommand(char c) {
  switch (c) {
    case 'H':
      Serial1.write('H');
      Serial.println("[DEBUG] Simulate: Close window");
      break;
    case 'T':
      Serial1.write('T');
      Serial.println("[DEBUG] Simulate: Earthquake alert");
      break;
    default:
      Serial.print("[DEBUG] Unknown command: ");
      Serial.println(c);
      break;
  }
```

- Created a UART-based Debug Mode on both MSP432 and Arduino
- Debug commands simulate real actions: lights, windows, doors, earthquake
- Arduino receives commands from MSP432 or PC Serial Monitor
- Red and white LEDs used to visually confirm command execution
- Earthquake logic includes blinking red LED and timeout system
- Helped test logic independently from physical sensors and buttons

CONCLUSIONS & FUTURE WORK

Università
degli Studi di
Trento

THANK YOU!

Islamele Landini

Sophie Motter

Tommaso Michelotti

Norris Kervatin