HANDY CURSOR

Introduction:

In an era driven by technological innovation, the convergence of augmented reality (AR) and gesture-based control has led to the creation of cutting-edge solutions for human-computer interaction. Among these innovations is the development of a "Handy Cursor," a novel approach that leverages the power of NodeMCU ESP8266, an MPU6050 accelerometer and gyroscope sensor, and a flex sensor to revolutionize the way we interact with our computer systems.

Augmented Reality, commonly known as AR, overlays digital information and interactive elements onto our real-world surroundings, enhancing our perception and interaction with the physical environment. In the context of "Handy Cursor," AR is utilized to create a virtual cursor that emulates the functions of a traditional computer mouse. This cursor is controlled not by a physical mouse or touchpad, but rather by hand movements in real-time.

The core components of this innovation include the NodeMCU ESP8266 microcontroller, which acts as the central processing unit, the MPU6050 sensor, which tracks and interprets hand movements in three-dimensional space, and the flex sensor, which detects hand gestures and translates them into mouse click actions. Together, these technologies create a seamless and intuitive means of interacting with a computer system.

The MPU6050 is responsible for tracking the orientation and acceleration of the user's hand, providing precise data on its position and movement. This information is relayed to the NodeMCU ESP8266, which processes the data and transforms it into cursor movements on the computer screen. By physically moving their hand in the air, users can effortlessly control the cursor's position, just as they would with a conventional mouse.

To replicate mouse click actions, the flex sensor is employed. This bendable sensor is strategically placed to detect the user's hand gestures. A gentle squeeze or release of the hand triggers a click, enabling users to perform actions like selecting, dragging, or clicking within the augmented reality environment.

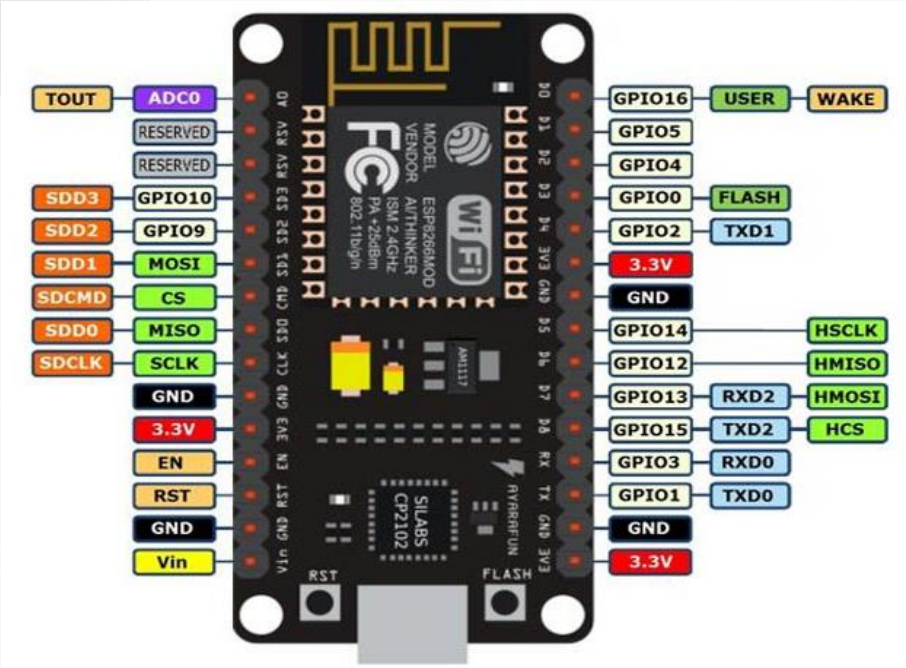
In conclusion, the Handy Cursor is a pioneering application of augmented reality and sensor technology that ushers in a new era of intuitive and hands-free computer interaction. By eliminating the need for physical peripherals, it offers an exciting glimpse into the future of human-computer interaction.

COMPONENTS REQUIRED:

1. NodeMCU ESP8266
2. MPU6050 (Accelerometer & Flex Sensor) Module
3. Flex Sensor
4. Bread Board
5. Jumper Wires
6. Resistor
7. Power supply
8. Glove
9. Arduino Ide
10. C and Python Programming Language

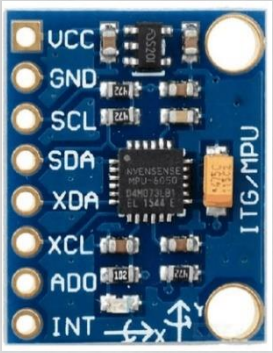
NodeMCU ESP8266:

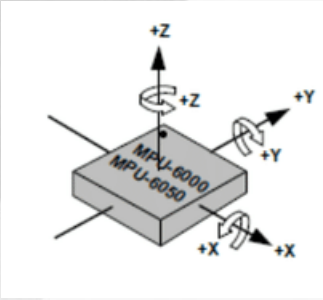
- The NodeMCU ESP8266 is a Wi-Fi enabled microcontroller that can be programmed using Lua, Arduino IDE, or MicroPython. It has 17 GPIO pins, 4 MB of flash memory, and 128 KB of RAM.
- It can be used for IoT projects that involve sensors, actuators, web servers, cloud services, and remote control. The most common version is v1.0, which has an ESP-12E chip.



Pin Category	Name	Description
Power	Micro-USB, 3.3V, GND, Vin	**Micro-USB:** NodeMCU can be powered through the USB port **3.3V:** Regulated 3.3V can be supplied to this pin to power the board **GND:** Ground pins **Vin:** External Power Supply
Control Pins	EN, RST	The pin and the button resets the microcontroller
Analog Pin	A0	Used to measure analog voltage in the range of 0-3.3V
GPIO Pins	GPIO1 to GPIO16	NodeMCU has 16 general purpose input-output pins on its board
SPI Pins	SD1, CMD, SD0, CLK	NodeMCU has four pins available for SPI communication.
UART Pins	TXD0, RXD0, TXD2, RXD2	NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program.
I2C Pins		NodeMCU has I2C functionality support but due to the internal functionality of these pins, you have to find which pin is I2C.

MPU6050 Module:



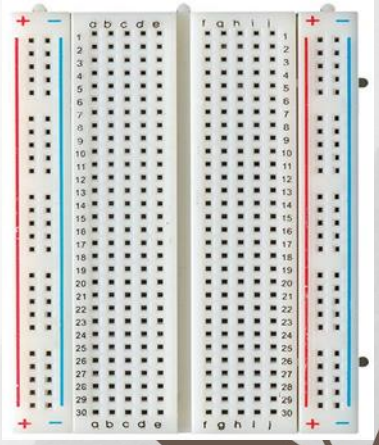


- The MPU6050 is a sensor module that combines a 3-axis accelerometer and a 3-axis gyroscope. It can measure motion parameters such as acceleration, velocity, orientation, and displacement.
- It also has a built-in temperature sensor and a digital motion processor (DMP) that can perform complex calculations. It communicates with a microcontroller using the I2C protocol and can be configured to have different I2C addresses.
- It is commonly used for projects that involve motion control, such as robots, drones, or self-balancing vehicles.

Flex Sensor:

- A flex sensor is a type of variable resistor that changes its resistance value when it is bent or flexed. The more the sensor is bent, the higher the resistance becomes.
- Flex sensors can be used to measure the angle or curvature of a surface, such as a finger, a glove, or a robot arm.
- Flex sensors are usually made of a flexible strip of conductive ink and material that can be attached to any surface

Bread Board:



- A breadboard is a reusable construction base used to build semi-permanent prototypes of electronic circuits. It has several holes where wires and electrical parts can be quickly placed without the need for soldering or destruction of tracks.
- Breadboards come in different sizes and types, such as solderless breadboards, protoboards, and power supply breadboards. They are commonly used in electronics projects for prototyping and testing circuits

Jumper Wires:

- Jumper wires are electrical wires with connector pins at each end. They are used to connect two points in a circuit without soldering. You can use jumper wires to modify a circuit or diagnose problems in a circuit.
- They come in a wide array of colors and types, such as male-to-male, male-to-female, and female-to-female. The wire color is just an aid to help you keep track of what is connected to which. It will not affect the operation of the circuit.

Resistor:

- A resistor is a component that reduces the flow of electric current in a circuit. It has two terminals that can be connected to other parts of the circuit. The resistance of a resistor is measured in ohms (Ω) and can be fixed or variable.
- Resistors are used for many purposes, such as dividing voltages, adjusting signal levels, biasing active elements, and terminating transmission lines

Power Supply:

- Lithium batteries are a type of battery that use lithium as the main component. They have high energy density, long lifespan, and low self-discharge rate. They are widely used in IoT devices, such as electric vehicles, sensors, and smart appliances.
- They can provide stable and reliable power supply for IoT devices that need to operate for long periods of time or in harsh environments.

Gloves:

- Gloves are used to IOT components such as bread board, power supply and flex sensor to embed on our hands.
- It is used to track the movement of our hand by embed the MPU6050 module on the gloves.
- Here we use gloves as cotton cloth which give better experience to wear it.

Arduino IDE:

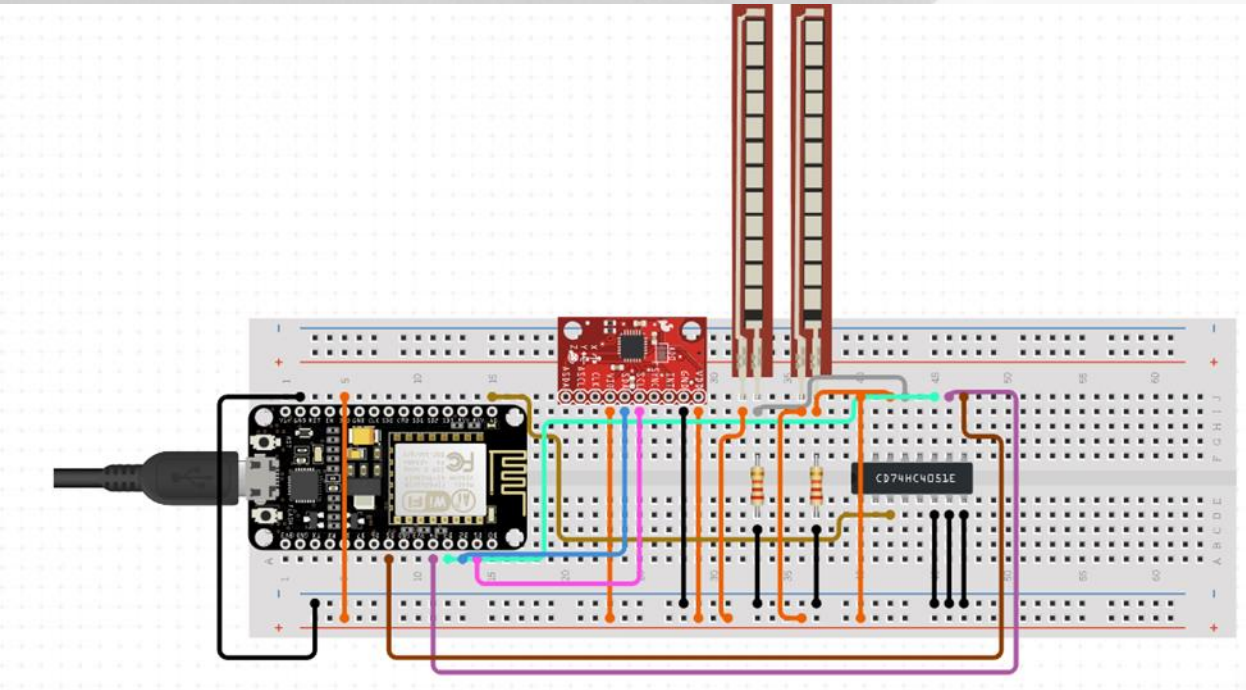
- Arduino IDE is a software that allows you to write, compile and upload code to Arduino boards and modules. It is open-source, easy-to-use and has many features such as autocompletion, code navigation and debugging.
- It supports various programming languages, such as C, C++, Python and MicroPython. You can download the latest version of Arduino IDE from their official website or use the web editor on Arduino Cloud.

C and Python Programming Language:

- C is a general-purpose programming language that is widely used in the field of embedded systems. The Arduino IDE supports programming in C++ and C languages.
- The C language is used to write low-level code that interacts directly with the hardware. In the Arduino IDE, you can write C code by creating a new file with a .c extension. You can then write your code using the standard C syntax and compile it using the Arduino IDE.
- Wire.h is a header file in the Arduino IDE that allows you to communicate with I2C/TWI devices . It provides methods for reading and writing data to and from I2C/TWI devices .
- Mouse.h is a header file in the Arduino IDE that provides methods for emulating a computer mouse . It includes methods for moving the mouse cursor, clicking the mouse buttons, and checking the status of the mouse buttons .

Experimental Approach:

Schematic Diagram:



Connection Process:

Hardware Connections:

Step 1. MPU6050 to NodeMCU ESP8266:

- Connect the MPU6050's VCC pin to the NodeMCU's 3.3V output.
- Connect the MPU6050's GND pin to the NodeMCU's GND.
- Connect the MPU6050's SDA pin to the NodeMCU's D2 (SDA) pin.
- Connect the MPU6050's SCL pin to the NodeMCU's D1 (SCL) pin.

Step 2. Flex Sensor to NodeMCU ESP8266:

- Connect one end of the flex sensor to the NodeMCU's 5V output (ensure it matches the sensor's voltage requirements).
- Connect the other end of the flex sensor to a 10kΩ resistor.
- Connect the other end of the 10kΩ resistor to the NodeMCU's A0 pin (analog input).
- Connect another 10kΩ resistor from the A0 pin to the NodeMCU's GND.

Software Setup:

Step 3.Arduino IDE:

- Open the Arduino IDE on your computer.

Step 4.Select Board and Port:

- In the Arduino IDE, go to "Tools" and select the appropriate board. Choose "NodeMCU 1.0 (ESP-12E Module)" or the equivalent board option for your NodeMCU ESP8266.
- Also, select the correct port where your NodeMCU is connected.

Step5.Upload the Code:

- Copy the provided code into the Arduino IDE.
- Click the "Upload" button (the right arrow) in the IDE to compile and upload the code to your NodeMCU ESP8266.

Step 6.Monitor Serial Output:

- Open the Serial Monitor in the Arduino IDE to monitor the output for debugging purposes. The code prints the Gyro and accelerometer values, which can help you understand how the sensor data affects mouse movement and mouse clicks.

Pin Connections:

NodeMCU ESP8266 Connections:

3.3V (VCC):

This is the power supply pin. Connect it to the VCC pin of the MPU6050. The NodeMCU operates at 3.3V, which is suitable for both the MPU6050 and the NodeMCU itself.

GND:

This is the ground pin. Connect it to the GND pin of the MPU6050 and the flex sensor to establish a common ground reference.

D2 (SDA):

This is the Serial Data (SDA) pin for I2C communication. Connect it to the SDA pin of the MPU6050.

D1 (SCL):

This is the Serial Clock (SCL) pin for I2C communication. Connect it to the SCL pin of the MPU6050.

5V (Flex Sensor):

Connect one end of the flex sensor to the 5V pin if your flex sensor requires a 5V supply. However, this is only applicable if your specific flex sensor operates at 5V. Ensure compatibility with your flex sensor's voltage requirements.

A0 (Flex Sensor):

Connect the other end of the flex sensor to the analog input A0. The NodeMCU's analog pins can measure voltage levels.

MPU6050 Connections:

VCC:

Connect it to the NodeMCU's 3.3V pin to power the MPU6050. Ensure that the MPU6050 is compatible with 3.3V operation.

GND:

Connect it to the NodeMCU's GND to establish a common ground reference.

SDA (Serial Data):

Connect it to the NodeMCU's D2 (SDA) for data communication in the I2C bus.

SCL (Serial Clock):

Connect it to the NodeMCU's D1 (SCL) for clock signals in the I2C bus.

Flex Sensor Connections:

One End to 5V:

Connect one end of the flex sensor to the NodeMCU's 5V pin if your flex sensor operates at 5V. Make sure this voltage matches the sensor's requirements.

Other End to 10kΩ Resistor:

The other end of the flex sensor connects to a 10kΩ resistor. This resistor helps create a voltage divider circuit to measure the flex sensor's resistance changes.

A0 (Analog Input):

Connect the junction between the flex sensor and the 10kΩ resistor to the NodeMCU's A0 pin. This pin reads the analog voltage produced by the voltage divider.

10kΩ Resistor to GND:

Connect another 10kΩ resistor from the A0 pin to the NodeMCU's GND. This ensures that the analog input is referenced correctly.

Code:

```
#include <Wire.h>
#include <Mouse.h>

const int MPU_addr = 0x68;
int16_t accelerometer_x, accelerometer_y, accelerometer_z;
int16_t gyroscope_x, gyroscope_y, gyroscope_z;

float Acc_x, Acc_y, Acc_z;
float Gyro_x, Gyro_y, Gyro_z;
float AccAngleX, AccAngleY;

const int flexPin = A0;
const int flexThreshold = 500;

void setup() {
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
  Serial.begin(9600);
  pinMode(flexPin, INPUT);
}
void loop() {
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr, 14, true);

  accelerometer_x = (Wire.read() << 8 | Wire.read());
  accelerometer_y = (Wire.read() << 8 | Wire.read());
  accelerometer_z = (Wire.read() << 8 | Wire.read());

  gyroscope_x = (Wire.read() << 8 | Wire.read());
  gyroscope_y = (Wire.read() << 8 | Wire.read());
  gyroscope_z = (Wire.read() << 8 | Wire.read());

  Acc_x = (float)accelerometer_x / 16384.0;
  Acc_y = (float)accelerometer_y / 16384.0;
  Acc_z = (float)accelerometer_z / 16384.0;

  AccAngleX = atan2(Acc_y, Acc_z) * 180 / PI;
  AccAngleY = atan2(Acc_x, sqrt(Acc_y * Acc_y + Acc_z * Acc_z)) * 180 / PI;

  int sensitivity = 10;
```

```
  int mouseX = map(Gyro_y, -1000, 1000, -sensitivity, sensitivity);
  int mouseY = map(Gyro_x, -1000, 1000, -sensitivity, sensitivity);

  Mouse.move(mouseX, mouseY);
  int flexValue = analogRead(flexPin);

  if (flexValue < flexThreshold) {
    Mouse.press(MOUSE_LEFT);
    delay(50); // Adjust the delay as needed
    Mouse.release(MOUSE_LEFT);
  }
  Serial.print("Gyro_x: ");
  Serial.print(Gyro_x);
  Serial.print(" Gyro_y: ");
  Serial.print(Gyro_y);
  Serial.print(" AccAngleX: ");
  Serial.print(AccAngleX);
  Serial.print(" AccAngleY: ");
  Serial.println(AccAngleY);
}
```

Code Explanation:

Include Libraries:

```
#include <Wire.h>

#include <Mouse.h>
```

These lines include two libraries: Wire for I2C communication and Mouse for emulating mouse actions. Wire is used to communicate with the MPU6050 sensor via I2C, and Mouse is used to control the mouse cursor.

Define Constants:

```
const int MPU_addr = 0x68;  // I2C address of MPU6050

int16_t accelerometer_x, accelerometer_y, accelerometer_z;

int16_t gyroscope_x, gyroscope_y, gyroscope_z;

float Acc_x, Acc_y, Acc_z;

float Gyro_x, Gyro_y, Gyro_z;

float AccAngleX, AccAngleY;

const int flexPin = A0;

const int flexThreshold = 500;
```

These lines define various constants and variables that will be used in the code. Notable constants include the I2C address of the MPU6050, pins for the flex sensor (A0) and the flex threshold (the value at which a mouse click is simulated).

Setup Function:

```
void setup() {
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
  Serial.begin(9600);
  pinMode(flexPin, INPUT);
}
```

- In the setup() function:
- The Wire library is initiated for I2C communication.
- The MPU6050 is initialized by sending a command to its Power Management 1 register to wake it up.
- Serial communication is started at a baud rate of 9600 for debugging.
- The flex sensor pin is configured as an input.

Loop Function:

```
void loop() {
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr, 14, true);
```

- In the loop() function:
- It communicates with the MPU6050 to read sensor data.
- It requests data from the MPU6050 starting from register 0x3B, which is the accelerometer data.

Read Sensor Data:

```
  accelerometer_x = (Wire.read() << 8 | Wire.read());
  accelerometer_y = (Wire.read() << 8 | Wire.read());
  accelerometer_z = (Wire.read() << 8 | Wire.read());
```

```
gyroscope_x = (Wire.read() << 8 | Wire.read());
gyroscope_y = (Wire.read() << 8 | Wire.read());
gyroscope_z = (Wire.read() << 8 | Wire.read());
```

Sensor data, including accelerometer and gyroscope readings, is read from the MPU6050. These values represent the orientation and movement of the sensor.

Calculate Angle and Map Gyroscope Data:

```
Acc_x = (float)accelerometer_x / 16384.0;
Acc_y = (float)accelerometer_y / 16384.0;
Acc_z = (float)accelerometer_z / 16384.0;
AccAngleX = atan2(Acc_y, Acc_z) * 180 / PI;
AccAngleY = atan2(Acc_x, sqrt(Acc_y * Acc_y + Acc_z * Acc_z)) * 180 / PI;
int sensitivity = 10;
int mouseX = map(Gyro_y, -1000, 1000, -sensitivity, sensitivity);
int mouseY = map(Gyro_x, -1000, 1000, -sensitivity, sensitivity);
```

- The code calculates angles (AccAngleX and AccAngleY) based on accelerometer data.
- It maps gyroscope data (Gyro_x and Gyro_y) to mouse cursor movement, allowing the sensor to control the cursor's position.
- The sensitivity variable controls the speed of cursor movement.

Mouse Control and Flex Sensor Check:

```
Mouse.move(mouseX, mouseY);
int flexValue = analogRead(flexPin);
if (flexValue < flexThreshold) {
  Mouse.press(MOUSE_LEFT);
  delay(50); // Adjust the delay as needed
  Mouse.release(MOUSE_LEFT);
 }
}
```

- The code uses the Mouse library to control the mouse cursor's movement.
- It reads the analog value from the flex sensor and checks if it's below the defined flex threshold. If so, it simulates a left mouse click by pressing and releasing the left mouse button using `

Experimental Result and Discussion:

Experimental Results:

Mouse Cursor Control:

The code is designed to use data from the MPU6050 accelerometer to control the movement of the mouse cursor. As you tilt or move the NodeMCU ESP8266, the MPU6050 will provide sensor data that gets mapped to mouse cursor movement. The experimental result should show that tilting or moving the sensor affects the cursor's position on the screen.

Flex Sensor Mouse Clicks:

The flex sensor is used to simulate mouse clicks. When you flex the sensor beyond the defined flexThreshold, the code will simulate a left mouse click. The experimental result will demonstrate that the sensor can control mouse clicks based on its flexing.

Serial Output:

The code provides serial output to the Arduino IDE's Serial Monitor for debugging and monitoring. You'll see the Gyro_x and Gyro_y values, which represent the movement of the sensor. Additionally, you'll see the calculated AccAngleX and AccAngleY, which represent the orientation of the sensor.

Discussion:

Accuracy and Sensitivity:

The accuracy of the mouse cursor control and flex sensor's responsiveness depends on several factors, including the calibration of the MPU6050 and the sensitivity settings in the code. You may need to fine-tune these parameters to achieve the desired level of accuracy and responsiveness.

Sensor Calibration:

The MPU6050 sensor should be properly calibrated to provide accurate data. Calibration is essential to ensure that the sensor's resting position corresponds to the cursor's center. You may need to implement sensor calibration routines if you find that the cursor doesn't return to the center when the sensor is at rest.

Flex Sensor Threshold:

The flex sensor's flex threshold value defines the point at which a mouse click is triggered. The threshold should be set so that it's easy to trigger a click when needed, but not so sensitive that unintentional clicks occur. You may need to adjust this threshold based on the specific characteristics of your flex sensor.

Environmental Conditions:

The experimental results may vary depending on environmental conditions and sensor placement. Make sure the sensor is placed on a stable surface and that there is minimal interference from vibrations or other external factors.

Real-world Applications:

This project can serve as a basis for creating unique interfaces and human-computer interaction devices. The ability to control the mouse cursor with physical movement and gestures can be applied in various applications, such as assistive technology, gaming, and creative projects.

Conclusion:

In conclusion, the Handy Cursor, a groundbreaking integration of augmented reality, sensor technology, and the NodeMCU ESP8266 microcontroller, represents a significant leap forward in the realm of human-computer interaction. This innovative solution reimagines the way we control our computer systems by replacing traditional input devices with intuitive, gesture-based control.

The Handy Cursor is not only a testament to the potential of augmented reality but also a promising avenue for improved accessibility. It has the potential to benefit a wide range of users, including those with mobility impairments, by providing a more natural and versatile means of computer interaction.