Design and Implementation of a Low-Cost Motion Controller Using Development Boards

[1]Yash Anand Ghadigaonkar , [2]Viraj Sameer Manjarekar , [3]Apurva Vinayak Malvankar, [4]Nirjara Sukhadev Jadhav
[1]Student , [2]Student ,[3]Student , [4]Student

[1234]Department of Electronics and Computer Science.
[1234]SSPM College of Engineering, Maharashtra, India.
yashanandghadigaonkar1@gmail.com,virajmanjarekar682@gmail.com,apurvamalvankar991@gmail.com,
nirjarajadhav766@gmail.com

Abstract **- Augmented Reality need input devices that are easy to use and quick to respond. Such devices should work for many different tasks. Normal tools like mouse and keyboard do not work well for immersive interaction [1]. AR controllers sold in market are costly and cannot be changed by users [2]. A controller was developed using Arduino Uno and MPU6050 sensor. Python was used to make it work with computer. The system work well without much delay. Sensor data can be mapped to any action required by the user. Cheap hardware and free software can be used together to make good solutions for AR, teaching labs, and games.**

Index Terms**- Arduino, MPU6050, development board, motion controller, Python interface**.

I. INTRODUCTION

Augmented Reality is now used for seeing things in new ways, for learning by doing, and for interactive teaching [1]. AR controllers sold in shops cost too much money. Users cannot change how they work [2]. For students, researchers, and people who like to make things at home there is a need for a controller that does not cost much, is easy to build, and can be used for different things [3, 4]. This paper addresses this problem. A design is presented using development boards like Arduino Uno and MPU6050 sensor. Python is used to map actions in a flexible way.

II. RELATED WORK

Arduino has been used for hand movement control in robots and automation [5, 6]. IMU sensors were used by some researchers to capture hand movements [1]. These systems work but they do not connect well with computer programs like AR.AR controllers from companies like HoloLens Clicker work fine but users cannot change anything inside them [2].They also cost too much for many colleges [3].The work presented here is different because open-source hardware and software are combined. A flexible interface is provided. Focus is on motion control using development boards. Recent work on open-source controllers [3, 4] shows interest in this area. Not many solutions give both motion sensing and software mapping that users can change.

III. SYSTEM ARCHITECTURE

The system has three parts: the sensor part, the processing part, and the software part. Each part does its own job. Movement and button data is captured, processed, and turned into actions. Hardware connections are kept simple. Software is kept clear too. This makes the system easy to use while maintaining good function.

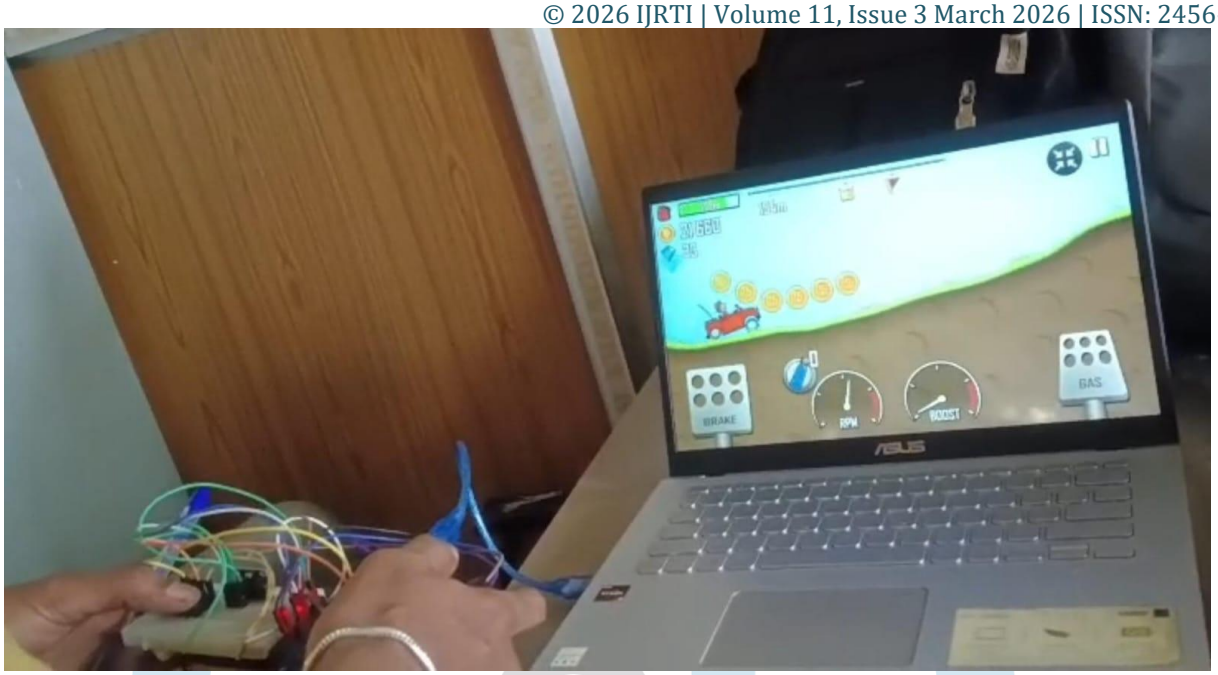

Figure 1: Physical prototype of the motion controller showing Arduino Uno, MPU6050 sensor, joystick and push buttons

A. Subsystem Overview

1) Sensor Part: This have MPU6050 sensor, a joystick, and push buttons. MPU6050 gives gyroscope and accelerometer data. This helps track movement. Joystick gives two-direction analog input. Buttons give on-off signals. Together they form the main input for the controller.

2) Processing Part: Arduino Uno is the main unit hear. Sensor data is taken using I2C. Data is cleaned by filtering and dead-zone setting. Then data is sent through USB cable.

3) Software Part: A Python program with Tkinter gives the user a window to work with. Serial data is received, sensor values are readed, and mapping to actions like keyboard presses is done. Users can change mappings without touching the hardware. This provide flexibility.

B. Block Diagram

The full system design is shown in Fig. 2. The picture show how data moves from sensors to Arduino and then to software that talks to the target program.

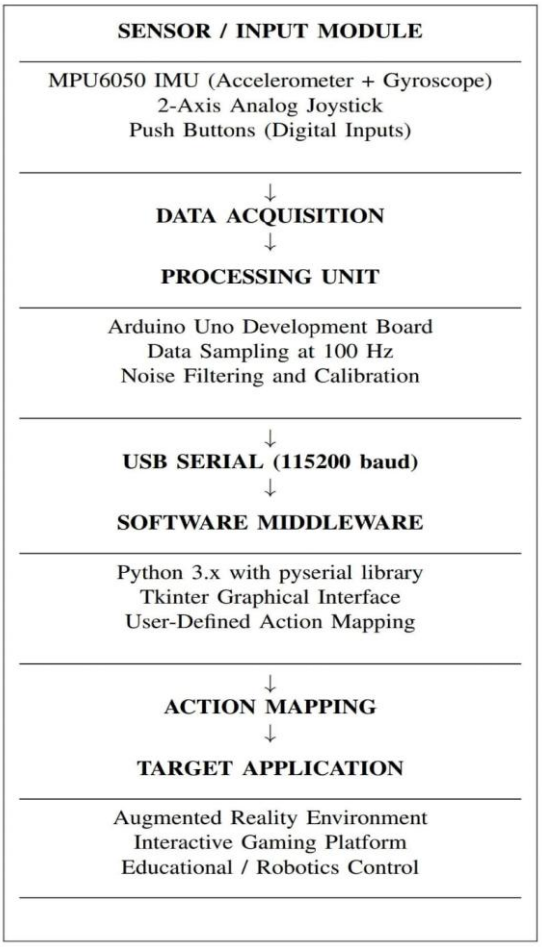

Figure 2: System architecture showing data flow between main layers

C. Components and Libraries

Hardware components were chosen for accessibility. Arduino Uno development board is widely available. Documentation is extensive. Beginners and advanced users can work with it. MPU6050 has both accelerometer and gyroscope in one package. Wiring is simple. Joystick and buttons cost little but provide good control.

For software, Arduino IDE is used for programming the development board. Libraries like Wire.h for I2C and MPU6050. h for sensor handling are utilized. The Python interface use pyserial for data reading. Tkinter provides a simple window interface. These tools are small and well-documented. The system can be copied and changed by others with little work.

The architecture uses simple hardware with free software libraries. A controller that work well and can be changed is obtained. Each part can be understood and changed by itself. This helps in teaching and research.

IV. METHODOLOGY

Systematic design, calibration, and testing was done for the controller. To begin with the MPU6050 sensor was set up using Wire.h library. I2C communication was established. Calibration was did to get correct accelerometer and gyroscope readings [1]. Drift and noise were reduced. Joystick dead-zone was set to avoid small unwanted movements. Buttons were debounced in software for reliable input.

After that Arduino Uno was coded to take sensor data 100 times each second. Data was packed as comma separated values: D1,D2,D3,D4,A1X,A2Y. D1 to D4 are digital inputs with values 0 or 1. A1X and A2Y are analog inputs that change continuously. This CSV format helped find user speed. Graphs could be plotted. Random key presses could be detected. Data was sent through USB.

Later on the computer, Python program used pyserial to read data continuously. Tkinter gave a window for action mapping. For example tilt left could become 'A' key press. Cursor position was calculated from analog values. Air mouse was added as an example. Waving the controller moves cursor like a mouse in air.

At the final stage testing was did in a simulated AR setup. Delay, data loss, and response speed were measured. Each part was tested alone first. Then all parts were put together. A controller suitable for motion control was obtained.

A. Hardware Design

Arduino Uno development board is the central unit. It talk to MPU6050 using I2C. MPU6050 is used by many for motion capture because it has both sensors [1, 5]. Following methods from [1], the sensor gives six-axis data at 100 Hz. Joystick gives two-direction input. Buttons give on-off actions like in [6].

B. Software Implementation

Arduino code use Wire.h for I2C. MPU6050. h is used for sensor data. Methods from [1] and [5] were followed. Data cleaning includes filtering and dead-zone as in [1]. Python use pyserial for serial communication. Real-time data processing is done. User actions are mapped. This follows the accessible design from [6].

1. Data Flow

Sensor data is collected at 100 Hz. Arduino processes it. Transmission is via USB at 115200 baud. Data format is CSV: D1,D2,D3,D4,A1X,A2Y. D1-D4 are digital (0 or 1). A1X and A2Y are analog (continuous). This format help find user speed. Graphs can be made. Random key presses can be seen. Python read this data. Input events are generated. Cursor position comes from analog values. Air mouse was added. Waving moves cursor like an air mouse.

V. WORKING PRINCIPLE

The controller work in four steps. These steps repeat very fast.

At the starting point sensors begin working. MPU6050 detects tilt or movement. Angle and force is measured [1]. Joystick tracks its two directions. Buttons wait for presses. All sensors send data continuously while controller is used.

Following this data go to Arduino Uno development board. Arduino acts as controller brain. Information is cleaned a little. If joystick is centered but shows tiny movement, deadzone removes that error. Random noise from MPU6050 is filtered. Movement readings become smooth. After cleaning, data is packed and sent to computer through USB. This happen 100 times each second. Data format is CSV: D1,D2,D3,D4,A1X,A2Y.

Then Python program on computer receive this data. The program translates incoming numbers. Rules can be set through a simple window. For example, tilt left make computer think 'A' key was pressed. Joystick forward moves mouse cursor up. This mapping make controller flexible. No hardware changes are needed. Only rules are changed. Cursor coordinates come from analog values. Air mouse was added. Waving controller moves cursor like an air mouse.

At the end mapped actions go to whatever app is running. This could be AR with 3D objects, a game, or other software. The app responds right away. Tilt make virtual object rotate. Joystick make game character walk.

The whole process take 20 to 45 milliseconds. This is fast enough. No delay is noticed by user. Movement feel natural on screen. The loop run 100 times per second. All movements and presses are caught. No inputs are missed.

VI. RESULTS AND DISCUSSION

The prototype was tested in a simulated environment. Responsiveness, accuracy, and ease of use were checked. During tests, smooth joystick movement was seen. Tilt detection was precise [1]. Button presses were reliable. MPU6050 with Arduino gave steady motion tracking. Python mapped sensor data to user actions correctly.

A. Performance

Delay is very important for interactive systems. Measurements showed average delay between 20 ms and 45 ms. This changed based on action complexity and sensor update speed. This delay is acceptable for real-time applications. Delays under 50 ms are not noticed by users. Data loss was very small. Communication between Arduino and Python stayed stable for long periods. Joystick gave linear response across its range. Dead-zone removed noise well. MPU6050 gave steady readings. Tilt movements were found without drift over time [1].

System was also checked under stress. Data sent non-stop for over two hours. No buffer overflow or errors happened. Python handled many inputs at once without freezing. Softwares design is robust. The prototype can handle interactive tasks with good accuracy and speed.

B. Cost Analysis

Comparison with commercial controllers show cost savings. Shop devices use special parts. Users cannot change them [2]. They cost more. This design use open hardware like Arduino development board and free software. Students and researchers can afford it.

Table 1: Cost Comparison Between Commercial Devices and Proposed System

Component	Commercial Device	Proposed System
Microcontroller	Proprietary	Arduino Uno
Motion Sensor	Integrated	MPU6050
Software	Closed	Python (open-source)
Total Cost (INR)	1700+	800

Less cost does not mean less function. The prototype works as fast and accurate as shop devices. Users can change software however they want. This help in teaching labs with less money. Free tools let students try new things. The controller can be used for robots, games, or other uses.

Results show the controller balances cost and function well. Cheap parts with good software give good results for motion control applications.

VII. APPLICATIONS

The controller can be used in several areas:

- **Augmented Reality:** Move 3D objects in virtual world.
- **Gaming:** Use as plane or car controller for moderate speed games.
- **Education:** Show how sensors and code work.
- **Robotics:** Control robots or drones with hand movements for teaching labs.

VIII. CONCLUSION

Design and testing of a low-cost motion controller using development boards was presented in this paper. The system use cheap parts like Arduino Uno and MPU6050 sensor along with free open-source software. Python provides the interface between hardware and applications. All code and designs can be modified by users.

Response time of the controller is moderate. Measurements showed delay between 20 to 45 milliseconds. This is acceptable for moderate gaming applications like puzzle games, educational simulations, and strategy games. Fast-paced action games may require better response but for casual gaming the performance is sufficient. In robotics applications the moderate response works well for slow moving robots, educational robot arms, and basic drone control. High speed competitive applications would need faster systems but for learning and hobby purposes the speed is adequate.

The controller can be used in many areas. In augmented reality users can move 3D objects. In gaming it works as plane or car controller for moderate speed games. In education it shows how sensors and programming work together. In robotics it controls robots or drones with hand movements for teaching labs.

Future work include adding wireless communication like Bluetooth or WiFi. More sensors can be added for better tracking. Voice commands can be integrated for hands-free control. The design can be made smaller with custom PCB. Battery power can be added for portable use. Mobile app support can be developed. More testing with different applications will be done.

ACKNOWLEDGMENT

Thanks to the Department of Electronics and Computer Science Engineering, SSPM College of Engineering for support during this project.

References

[1] A. F. Panaite, M. Leba, M. L. Olar, R. C. Sibisanu, and L. Pellegrini, “Human arm motion capture using gyroscopic sensors,” in *MATEC Web of Conferences*, vol. 343, 2021.

[2] Y. J. Chen, W. C. Tung, W. R. Lee, et al., “Statistical analysis of noise measurement system based on accelerometer-gyroscope GY-521 and Arduino platform,” in *2017 IEEE International Conference on Applied System Innovation (ICASI)*, 2017, pp. 1234-1237.

[3] N. B. Ananta, C. Kimber, R. Du, and D. Kim, “RetroSphere: Enabling low-power and affordable 3D input for lightweight augmented reality devices,” in *GetMobile: Mobile Computing and Communications*, vol. 28, no. 2, pp. 31-37, 2024.

[4] M. Lee, S. Patel, and J. Chen, "Attina: An open-source low-cost optical see-through AR headset," in *2023 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp/ISWC '23)*, 2023.

[5] K. N. Pham, T. C. Nguyen, and H. M. Tran, "Real-time gesture recognition using MPU6050 and Arduino for human-computer interaction," in *2020 International Conference on Advanced Technologies for Communications (ATC)*, 2020, pp. 245-249.

[6] S. K. Pulipati, V. R. Nandanoori, and D. V. Ratnam, "Real-time mouse control system using hand gesture recognition with MPU6050 sensor," in *2020 International Conference on Trends in Electronics and Informatics (ICOEI)*, 2020, pp. 456-460.