SMART STROKE

TEAM MEMBERS:

MOUNIKA SATYA PRASEEDA KALLAKURI
SEGU SAHISHNA
SHREYA SHAMEJ
SAI SRI BHAVANA THUMMALA

UNDER THE GUIDANCE OF
DR. AMEET CHAVAN

ABSTRACT

In the real time of sports, the pursuit of excellence is an unending endeavor. The advent of technology has paved the way for innovative solutions to enhance athlete performance and training methodologies. This paper introduces the "Smart Bat," a groundbreaking sports equipment designed to bridge the gap between traditional sports tools and data-driven insights. By seamlessly integrating sensor technology, wireless connectivity, and real-time data analysis, the Smart Bat empowers athletes with a comprehensive tool to elevate their skills and refine their techniques.

The Smart Bat represents a transformative leap forward in sports technology, offering athletes the ability to receive immediate, objective feedback during practice and gameplay. This real-time data-driven approach fosters continuous improvement, enabling athletes to fine-tune their skills and refine their strategies. As a result, the Smart Bat has the potential to redefine the boundaries of sports performance, ushering in a new era of data-driven training and elevated gameplay

MOTIVATION

The motivation behind the development of the Smart Bat stems from a desire to revolutionize sports training and performance. Traditional sports equipment often lacks the ability to provide real-time, objective feedback to athletes during practice and gameplay. This limitation inspired the creation of the Smart Bat, which combines cutting-edge sensor technology with wireless connectivity to offer athletes valuable insights into their techniques. By bridging the gap between data and sports, the Smart Bat empowers athletes to make informed adjustments, refine their skills, and unlock their full potential. This innovation has the potential to reshape how athletes train, enhance their competitive edge, and ultimately redefine the standards of excellence in sports.

INDEX

1. **INTRODUCTION**
    a. Background
    b. Available solution
2. **BLOCK DIAGRAM**
3. **HARDWARE COMPONENTS**
4. **HARDWARE DESCRIPTION**
5. **METHODOLOGY**
    a. Installing Arduino IDE
    b. Working Mechanism
    c. Code
6. **RESULTS AND DISCUSSIONS**
7. **CONCLUSION**
8. **REFERENCES**

1. INTRODUCTION

A. BACKGROUND:

The emergence of smart bat analysis represents a paradigm shift in the world of sports, where technology converges with athletic equipment to revolutionize player performance and training methodologies. In recent years, sports equipment has evolved from basic tools to sophisticated instruments capable of capturing and processing intricate data in real time. This evolution has given rise to the concept of smart bat analysis, where cutting-edge sensors, data analytics, and biomechanics intersect to provide athletes, coaches, and analysts with unprecedented insights into the dynamics of each swing, impact, and movement.

The absence of an advanced smart bat analysis system poses several challenges that need to be addressed.

- Inadequate Data Accuracy: The existing methods fail to precisely measure and record crucial parameters such as bat speed, ball impact, swing trajectory.
- Player-Specific Adaptability: Each player possesses unique characteristics, body mechanics, and playing styles.
- Real-Time Insights and Feedback: The current methods lack real-time analysis capabilities, depriving players and coaches of immediate feedback during practice sessions and matches.

B. AVAILABLE SOLUTION/OBJECTIVE:

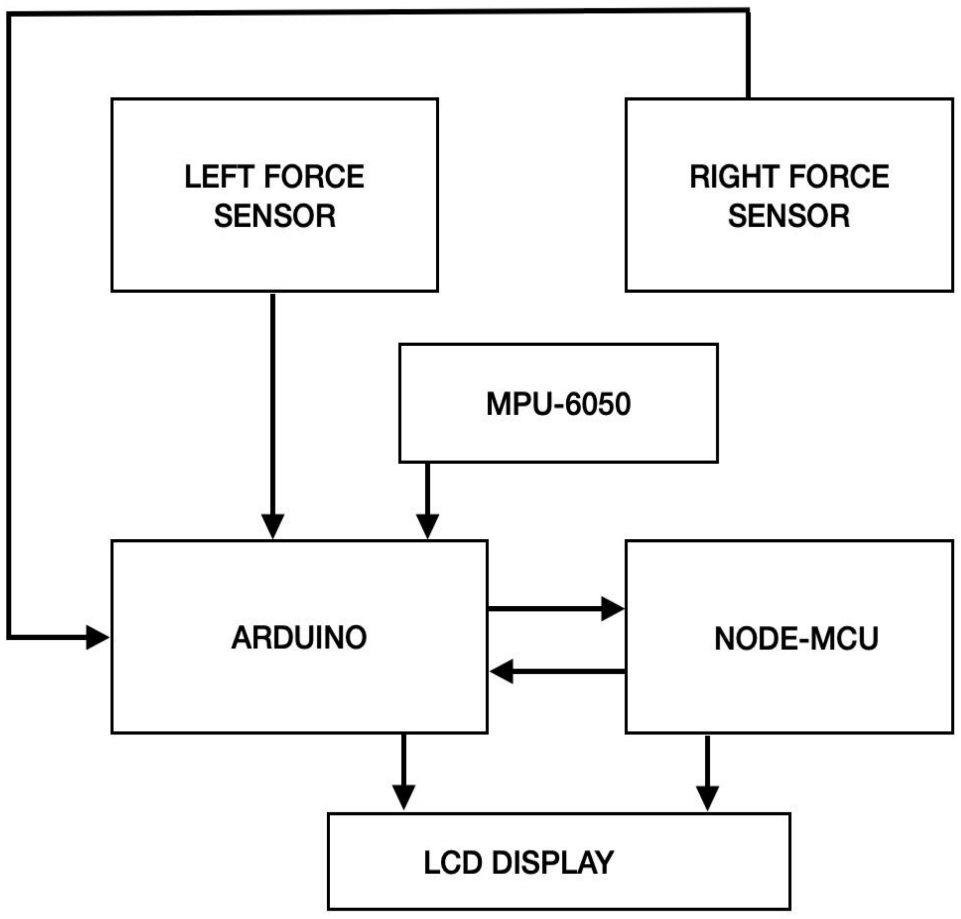
Beyond training, these intelligent bats enhance in-game performance by offering data-driven insights that aid players in adjusting their strategies based on factors like pitch speed and trajectory. In essence, smart bats signify a harmonious blend of technology and tradition, propelling the sports industry into a new era of data-driven excellence.

To overcome the challenges posed by the current methods of sports performance analysis, the development of smart bat analysis system is proposed.

To harness the power of advanced technology, data analysis, sensor technology, comparison with batting techniques and feedback mechanism.

This system aims to revolutionize sports performance assessment by providing players, and analysts with a comprehensive and objective understanding of batting techniques, strengths, weaknesses, and strategic decision-making.

2. BLOCK DIAGRAM

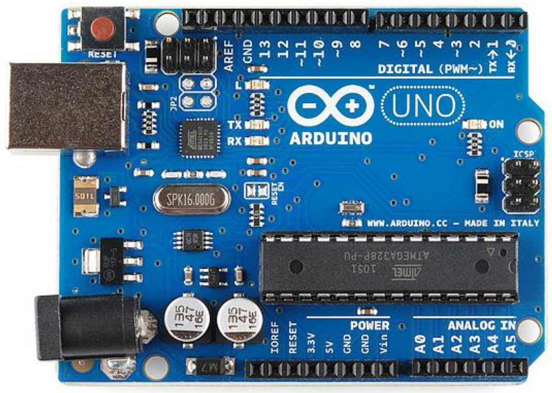


3. HARDWARE COMPONENTS:

1. Arduino
2. Node MCU
3. MPU6050
4. Force Sensor
5. LCD Display

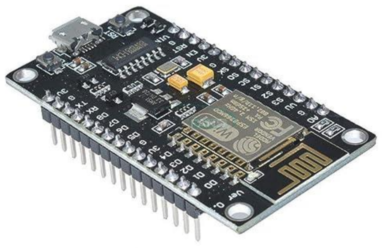
4. HARDWARE DESCRIPTION:

The Arduino: It is the central controller in the smart bat analysis project. It collects data from sensors (MPU6050, force sensor), processes swing metrics, provides real-time LCD feedback, enables wireless data transmission, integrates sensor data for comprehensive analysis, and allows for customization and expansion of functionalities. It serves as the core hub for data processing and feedback mechanisms.



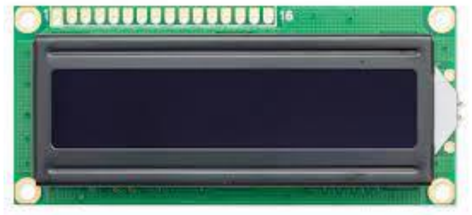
MPU6050: The MPU-6050 is not a processor itself, but rather a sensor module. It is a popular integrated circuit that combines a 3-axis accelerometer and a 3-axis gyroscope. It provides accurate measurements of acceleration and angular velocity along three axes, making it valuable for tasks such as motion tracking.

Node MCU: Node MCU is an open-source development board based on the ESP8266 WIFI module. It integrates a microcontroller with built-in WIFI, allowing easy IoT and prototyping projects. With Lua scripting support, it enables rapid development of web-connected applications.



Force Sensor: A force sensor is a device designed to measure the force applied to it. It converts mechanical force into an electrical signal that can be interpreted and analyzed by various instruments or systems.

LCD Display: An LCD (Liquid Crystal Display) is a flat-panel technology commonly used in screens for devices like TVs, monitors, and digital devices. It works by manipulating liquid crystals to produce images or text with high resolution and color accuracy. LCDs are energy-efficient and widely used in various applications, from consumer electronics to industrial displays.



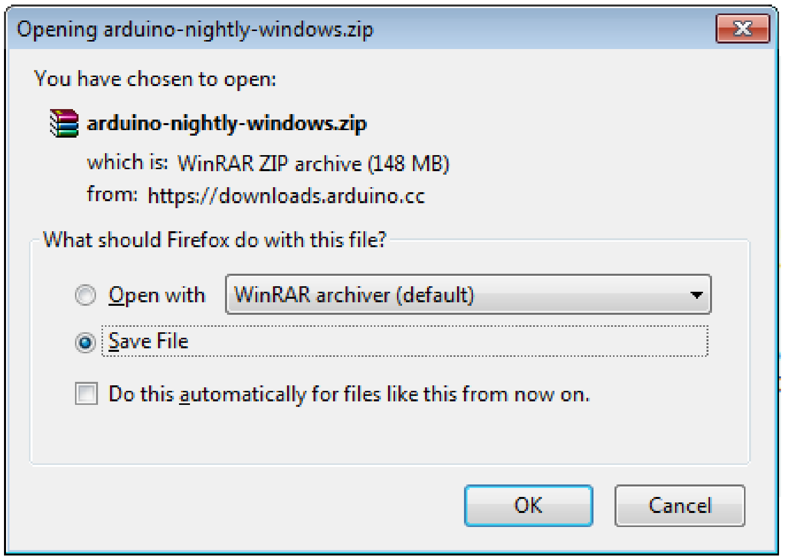
5. METHODOLOGY:

Installing Arduino Ide:

Step 1 − First you must have your Arduino board (you can choose your favorite board) and a USB cable.

Step 2 − Download Arduino IDE Software.

You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.


Opening arduino-nightly-windows.zip

You have chosen to open:

arduino-nightly-windows.zip

which is: WinRAR ZIP archive (148 MB)

from: https://downloads.arduino.cc

What should Firefox do with this file?

Open with: WinRAR archiver (default)

[✓ Save File]

Do this automatically for files like this from now on.

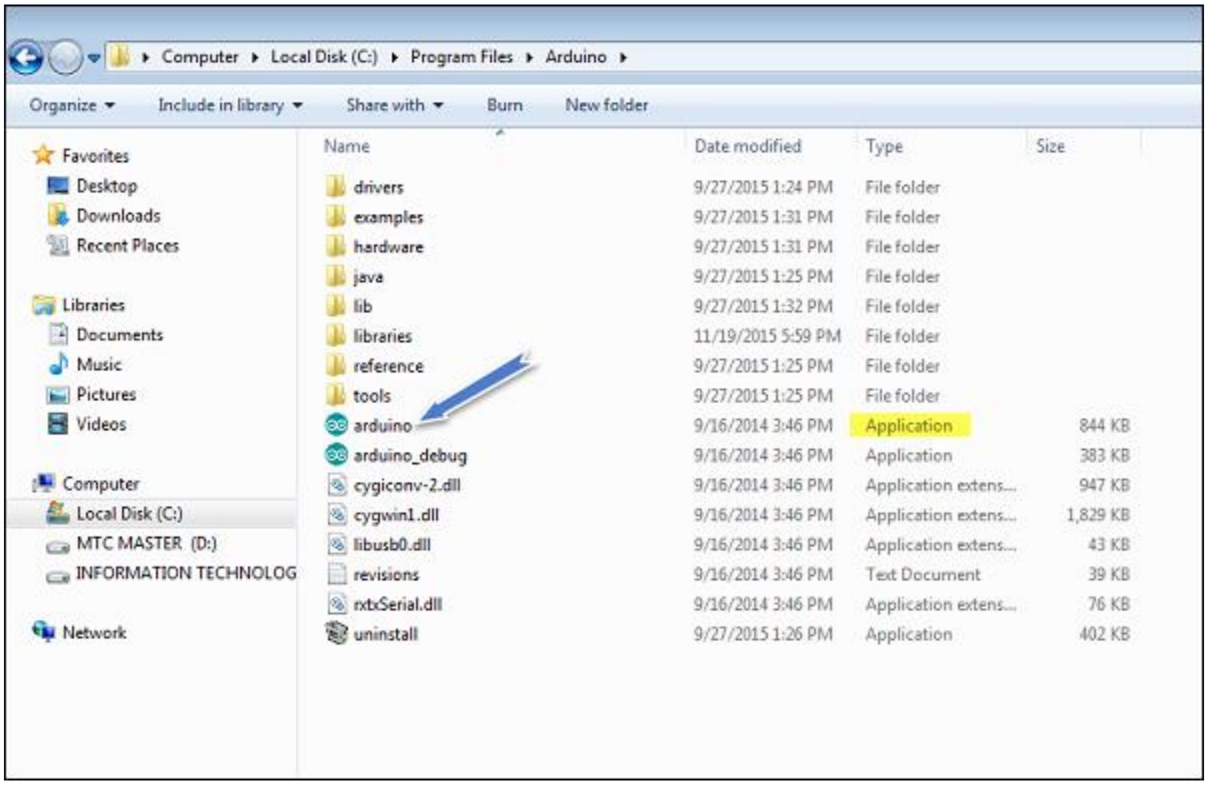
Step 3 − Power up your board.

The Arduino Uno and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

Step 4 − Launch Arduino IDE.

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.



Name	Date modified	Type	Size
drivers	9/27/2015 1:24 PM	File folder	
examples	9/27/2015 1:31 PM	File folder	
hardware	9/27/2015 1:31 PM	File folder	
java	9/27/2015 1:25 PM	File folder	
lib	9/27/2015 1:32 PM	File folder	
libraries	11/19/2015 5:59 PM	File folder	
reference	9/27/2015 1:25 PM	File folder	
tools	9/27/2015 1:25 PM	File folder	
arduino	9/16/2014 3:46 PM	Application	844 KB
arduino_debug	9/16/2014 3:46 PM	Application	383 KB
cygiconv-2.dll	9/16/2014 3:46 PM	Application extens...	947 KB
cygwin1.dll	9/16/2014 3:46 PM	Application extens...	1,829 KB
libusb0.dll	9/16/2014 3:46 PM	Application extens...	43 KB
revisions	9/16/2014 3:46 PM	Text Document	39 KB
rxtxSerial.dll	9/16/2014 3:46 PM	Application extens...	76 KB
uninstall	9/27/2015 1:26 PM	Application	402 KB

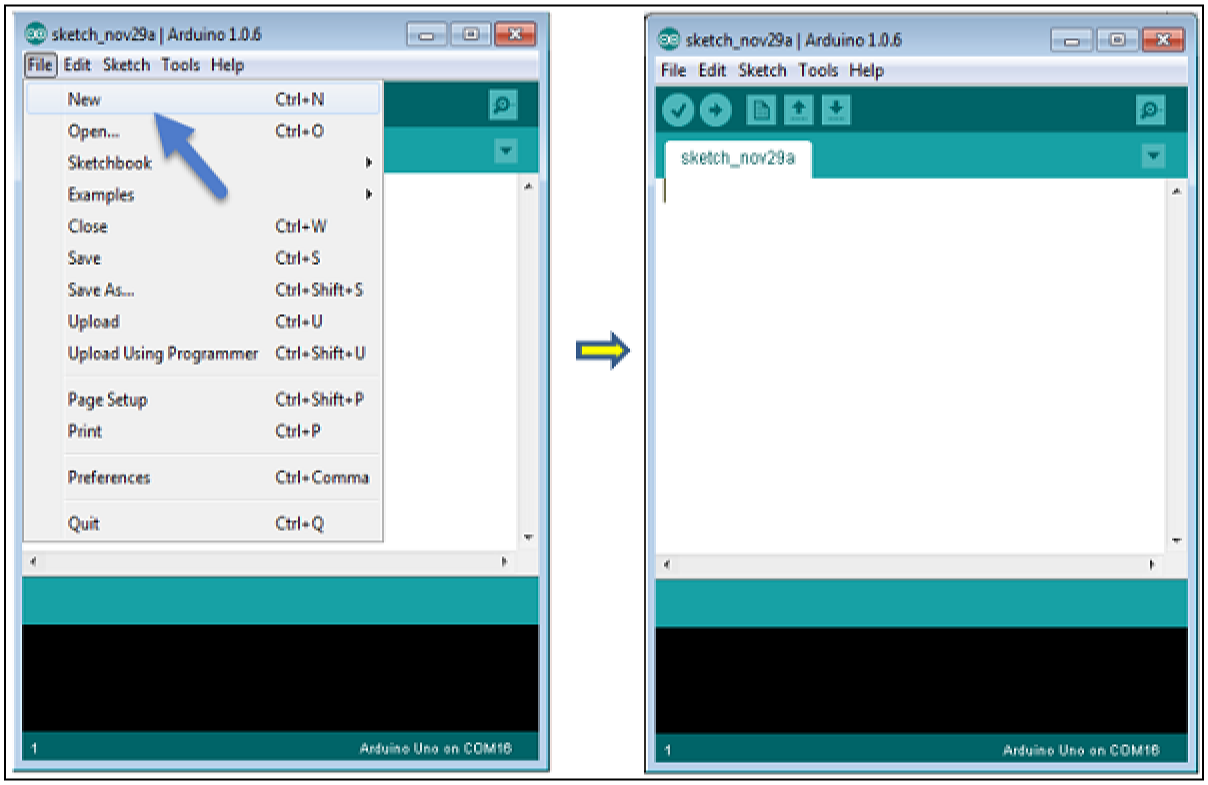
Step 5 − Open your first project.

Once the software starts, you have two options −

Create a new project.

Open an existing project example.

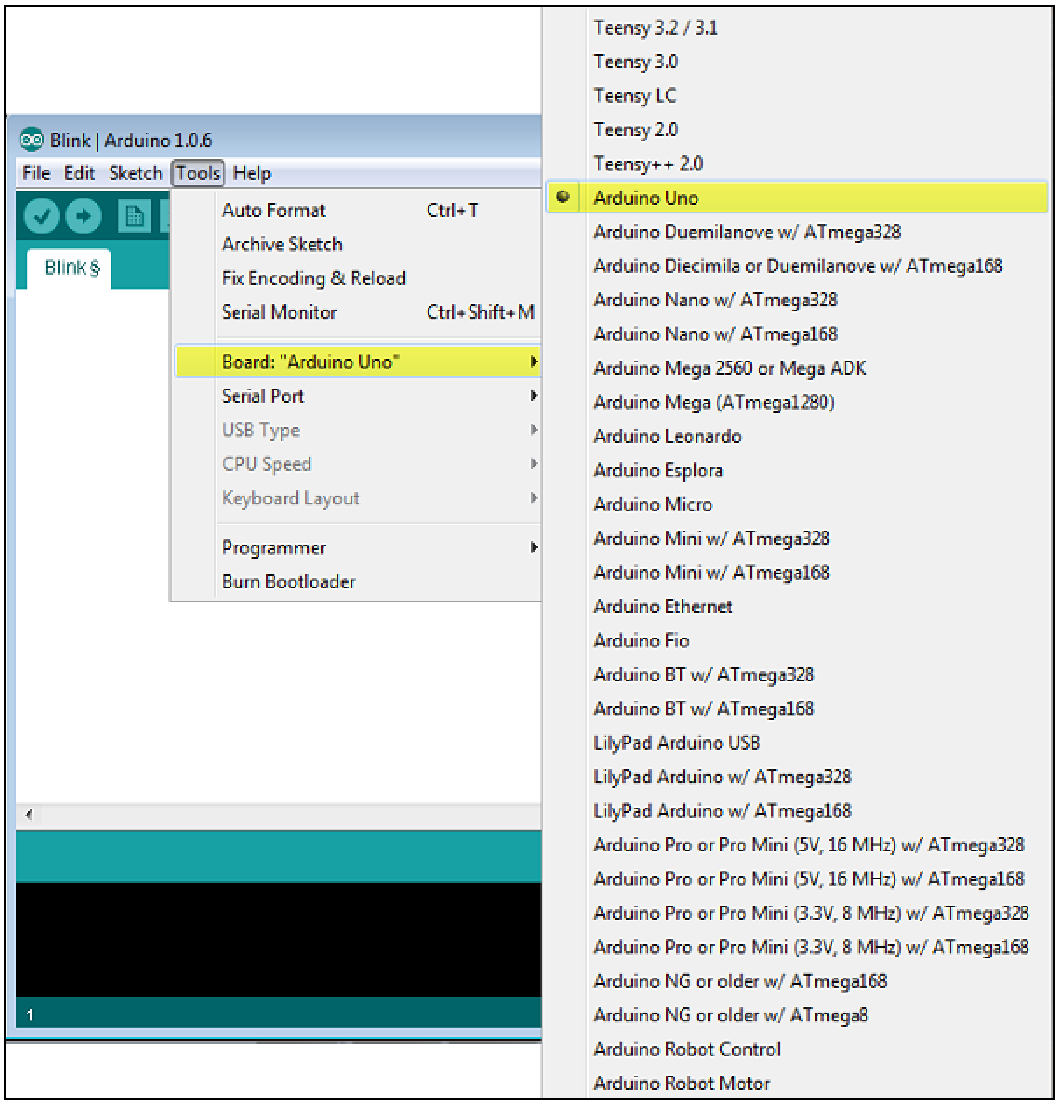
To create a new project, select File → New.



Step 6 − Select your Arduino board.

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

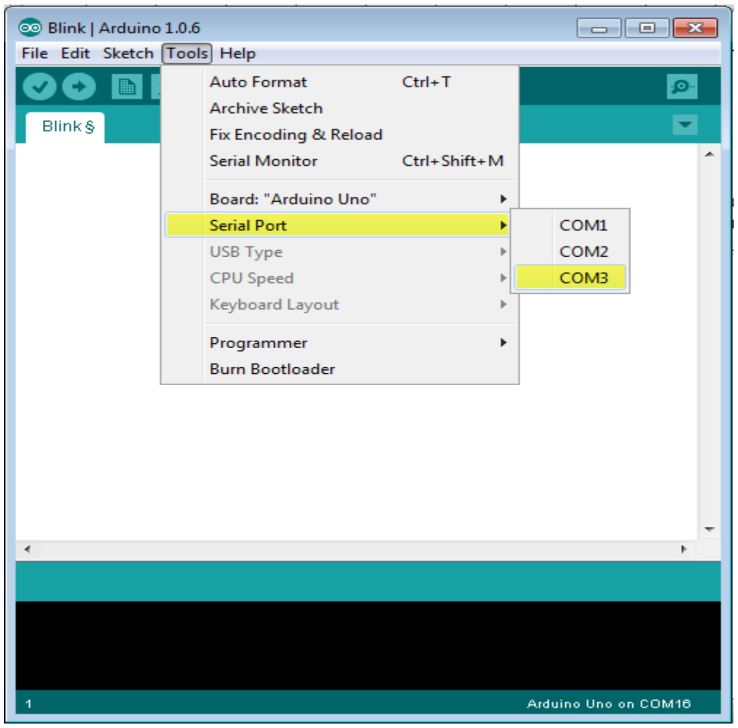
Go to Tools → Board and select your board.


Blink | Arduino 1.0.6
File Edit Sketch Tools Help
Blink §
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board: "Arduino Uno"
Serial Port
USB Type
CPU Speed
Keyboard Layout
Programmer
Burn Bootloader
1
Teensy 3.2 / 3.1
Teensy 3.0
Teensy LC
Teensy 2.0
Teensy++ 2.0
Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora
Arduino Micro
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino USB
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8
Arduino Robot Control
Arduino Robot Motor

Here, we have selected the Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using.

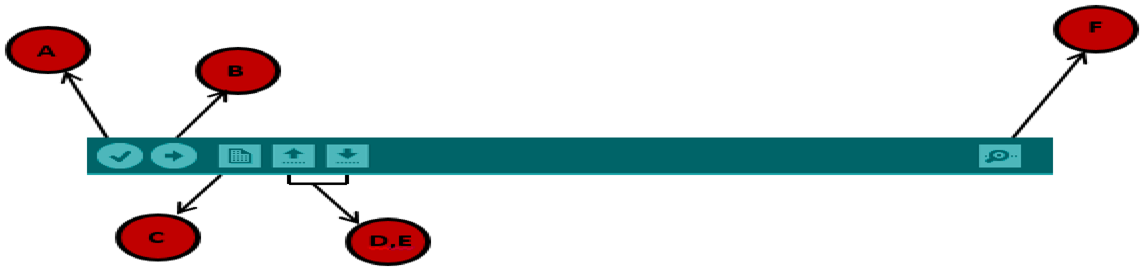
Step 7 − Select your serial port.

Select the serial device of the Arduino board. Go to Tools → Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.



Step 8 − Upload the program to your board.

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.



A − Used to check if there is any compilation error.

B − Used to upload a program to the Arduino board.

C − Shortcut used to create a new sketch.

D − Used to directly open one of the example sketch.

E − Used to save your sketch.

F − Serial monitor used to receive serial data from the board and send the serial data to the board.

Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

Note − If you have an Arduino Mini, NG, or other board, you need to press the reset button physically on the board, immediately before clicking the upload button on the Arduino Software.

WORKING MECHANISM:

A "smart bat" could refer to a cricket bat that incorporates technology to enhance performance analysis, training, or user experience. Here's a conceptual overview of what a smart bat might entail:

Features and Components:

1.Sensor Integration:

•Integrate the MPU6050 sensor to measure motion data (acceleration and gyroscope) and the force sensor to measure the force exerted during a bat swing.

2.Data Acquisition:

•Read acceleration and gyroscope data from the MPU6050 sensor to determine the orientation and angular velocity of the bat.

•Read force data from the force sensor to measure the left and right side forces during the ball hits the bat

3.Wireless Connectivity:

• wireless technology to transmit data from the bat to a connected device such as a smartphone or tablet.

4.Performance Tracking:

•The time and identify areas for improvement.bat could keep a record of each swing, allowing players to track progress over

5.Customizable Profiles:

•Users could create and save profiles with their preferences, skill level, and goals.

6.Remote Coaching:

Coaches could remotely monitor their players' performance and offer guidance based on the collected data.

7.Visualization:

•When a user swings the bat, the force data is collected and analyzed in real time.

•The LCD display shows the left and right side force values as well as the axis of bat projection, which could be represented as an arrow or numerical value indicating the axis.

8.Feedback and Analysis:

•Users can interpret the displayed information to understand their swing's force distribution and the angle of bat projection.

•The LCD provides real-time feedback, allowing users to make adjustments to their technique and improve their swing.

9.Research and Analysis:

Aggregated data from multiple users could be used for research purposes, helping to understand batting techniques and trends.

CODE:

```
// Basic demo for accelerometer readings from Adafruit MPU6050

#include <Adafruit_MPU6050.h>

#include <Adafruit_Sensor.h>

#include <Wire.h>

#include <LiquidCrystal.h>

#include <Adafruit_NeoPixel.h>

#ifdef _AVR_

 #include <avr/power.h> // Required for 16 MHz Adafruit Trinket

#endif

// Which pin on the Arduino is connected to the NeoPixels?

// On a Trinket or Gemma we suggest changing this to 1:

#define LED_PIN_l    6

#define LED_PIN_r    7

// How many NeoPixels are attached to the Arduino?

#define LED_COUNT  4


// Declare our NeoPixel strip object:

Adafruit_NeoPixel stripl(LED_COUNT, LED_PIN_l, NEO_GRB + NEO_KHZ800);

Adafruit_NeoPixel stripr(LED_COUNT, LED_PIN_r, NEO_GRB + NEO_KHZ800);

const int rs = 8, en = 9, d4 = 10, d5 =11, d6 = 12, d7 = 13;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
```

```
Adafruit_MPU6050 mpu;

int fs1=A2;

int fs2=A3;

void setup(void) {

  Serial.begin(115200);

  while (!Serial)

    delay(10); // will pause Zero, Leonardo, etc until serial console opens

  Serial.println("Adafruit MPU6050 test!");

  lcd.begin(16, 2);

  lcd.print("   WELCOME");

  stripl.begin();           // INITIALIZE NeoPixel strip object (REQUIRED)

  stripl.show();            // Turn OFF all pixels ASAP

  stripl.setBrightness(50);

    stripr.begin();           // INITIALIZE NeoPixel strip object (REQUIRED)

  stripr.show();            // Turn OFF all pixels ASAP

  stripr.setBrightness(50);

  // Try to initialize!

  if (!mpu.begin()) {

    Serial.println("Failed to find MPU6050 chip");

    while (1) {

      delay(10);

    }
```

```
}
Serial.println("MPU6050 Found!");
mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
Serial.print("Accelerometer range set to: ");
switch (mpu.getAccelerometerRange()) {
case MPU6050_RANGE_2_G:
  Serial.println("+-2G");
  break;
case MPU6050_RANGE_4_G:
  Serial.println("+-4G");
  break;
case MPU6050_RANGE_8_G:
  Serial.println("+-8G");
  break;
case MPU6050_RANGE_16_G:
  Serial.println("+-16G");
  break;
}
mpu.setGyroRange(MPU6050_RANGE_500_DEG);
Serial.print("Gyro range set to: ");
switch (mpu.getGyroRange()) {
case MPU6050_RANGE_250_DEG:
  Serial.println("+- 250 deg/s");
  break;
case MPU6050_RANGE_500_DEG:
  Serial.println("+- 500 deg/s");
  break;
case MPU6050_RANGE_1000_DEG:
  Serial.println("+- 1000 deg/s");
  break;
case MPU6050_RANGE_2000_DEG:
  Serial.println("+- 2000 deg/s");
  break;
}
mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
Serial.print("Filter bandwidth set to: ");
switch (mpu.getFilterBandwidth()) {
case MPU6050_BAND_260_HZ:
  Serial.println("260 Hz");
  break;
case MPU6050_BAND_184_HZ:
  Serial.println("184 Hz");
  break;
case MPU6050_BAND_94_HZ:
  Serial.println("94 Hz");
  break;
case MPU6050_BAND_44_HZ:
  Serial.println("44 Hz");
  break;
case MPU6050_BAND_21_HZ:
  Serial.println("21 Hz");
  break;
case MPU6050_BAND_10_HZ:
  Serial.println("10 Hz");
  break;
case MPU6050_BAND_5_HZ:
  Serial.println("5 Hz");
  break;
}
Serial.println("");
delay(100);
stripl.setPixelColor(0, stripl.Color(50,50,   50));         //  Set pixel's color (in RAM)
stripl.setPixelColor(1, stripl.Color(50,50,   50));
stripl.setPixelColor(2, stripl.Color(50,50,   50));
stripl.setPixelColor(3, stripl.Color(50,50,   50));
stripr.setPixelColor(0, stripr.Color(50,50,   50));         //  Set pixel's color (in RAM)
```

```
  stripr.setPixelColor(1, stripr.Color(50,50,   50));
  stripr.setPixelColor(2, stripr.Color(50,50,   50));
  stripr.setPixelColor(3, stripr.Color(50,50,   50));
  stripl.show();
  delay(200);
  stripr.show();
}
void loop() {
  /* Get new sensor events with the readings */
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);
  /* Print out the values */
  Serial.print("Acceleration X: ");
  Serial.print(a.acceleration.x);
  Serial.print(", Y: ");
  Serial.print(a.acceleration.y);
  Serial.print(", Z: ");
  Serial.print(a.acceleration.z);
  Serial.println(" m/s^2");
  Serial.print("   Rotation X: ");
  Serial.print(g.gyro.x);
  Serial.print(", Y: ");
  Serial.print(g.gyro.y);
  Serial.print(", Z: ");
  Serial.print(g.gyro.z);
  Serial.print(" rad/s");
  Serial.print("   Temperature: ");
  Serial.print(temp.temperature);
  Serial.print(" degC");
  int f1val=analogRead(fs1);
  int f2val=analogRead(fs2);
  Serial.print("   FS1: ");
  Serial.print(f1val);
  Serial.print("  FS2: ");
  Serial.println(f2val);
  delay(5);
  lcd.clear();
  lcd.print("X"+String(int(a.acceleration.x)) + " Y"+String(int(a.acceleration.y)) + "
Z"+String(int(a.acceleration.z)));
  lcd.setCursor(0,1);
  lcd.print("LF:"+String(f1val) + " RF"+String(f2val));
  if(f1val>10 || f2val>10)
  {
  if(f1val>10)
  {
```

```
  if(f1val>750)

  {

  stripl.setPixelColor(0, stripl.Color(255,0,   0));        //  Set pixel's color (in RAM)

  stripl.setPixelColor(1, stripl.Color(255,0,   0));

  stripl.setPixelColor(2, stripl.Color(255,0,   0));

  stripl.setPixelColor(3, stripl.Color(255,0,   0));

  }

    else if(f1val>500)

  {

  stripl.setPixelColor(1, stripl.Color(255,0,   0));        //  Set pixel's color (in RAM)

  stripl.setPixelColor(0, stripl.Color(255,0,   0));

  stripl.setPixelColor(2, stripl.Color(255,0,   0));

  stripl.setPixelColor(3, stripl.Color(50,50,   50));

  }

    else if(f1val>250)

  {

  stripl.setPixelColor(0, stripl.Color(255,0,   0));        //  Set pixel's color (in RAM)

  stripl.setPixelColor(1, stripl.Color(255,0,   0));

  stripl.setPixelColor(2, stripl.Color(50,50,   50));

  stripl.setPixelColor(3, stripl.Color(50,50,   50));

  }

else if(f1val>10)
```

```
 {
 stripl.setPixelColor(0, stripl.Color(255,0,   0));         //  Set pixel's color (in RAM)
 stripl.setPixelColor(1, stripl.Color(50,50,   50));
 stripl.setPixelColor(2, stripl.Color(50,50,   50));
 stripl.setPixelColor(3, stripl.Color(50,50,   50));
 }
  stripl.show();
  delay(200);
 }
if(f2val>10){
if(f2val>750) {
   Serial.println("R4");
 stripr.setPixelColor(0, stripr.Color(255,0,   0));         //  Set pixel's color (in RAM)
 stripr.setPixelColor(1, stripr.Color(255,0,   0));
 stripr.setPixelColor(2, stripr.Color(255,0,   0));
 stripr.setPixelColor(3, stripr.Color(255,0,   0));
 }
else if(f2val>500)
 {
   Serial.println("R3");
 stripr.setPixelColor(1, stripr.Color(255,0,   0));         //  Set pixel's color (in RAM)
 stripr.setPixelColor(0, stripr.Color(255,0,   0));
 stripr.setPixelColor(2, stripr.Color(255,0,   0));

 stripr.setPixelColor(3, stripr.Color(50,50,   50));

 }    else if(f2val>250)

 {

   Serial.println("R2");

 stripr.setPixelColor(0, stripr.Color(255,0,   0));         //  Set pixel's color (in RAM)

 stripr.setPixelColor(1, stripr.Color(255,0,   0));

 stripr.setPixelColor(2, stripr.Color(50,50,   50));

 stripr.setPixelColor(3, stripr.Color(50,50,   50));

 }

   else if(f2val>10)

 {

   Serial.println("R1");

 stripr.setPixelColor(0, stripr.Color(255,0,   0));         //  Set pixel's color (in RAM)

 stripr.setPixelColor(1, stripr.Color(50,50,   50));

 stripr.setPixelColor(2, stripr.Color(50,50,   50));

 stripr.setPixelColor(3, stripr.Color(50,50,   50));

 }

 stripr.show();

 }

 delay(2000);
```

```
   stripl.setPixelColor(0, stripl.Color(50,50,   50));         //  Set pixel's color (in RAM)

  stripl.setPixelColor(1, stripl.Color(50,50,   50));

  stripl.setPixelColor(2, stripl.Color(50,50,   50));

  stripl.setPixelColor(3, stripl.Color(50,50,   50));

  stripr.setPixelColor(0, stripr.Color(50,50,   50));         //  Set pixel's color (in RAM)

  stripr.setPixelColor(1, stripr.Color(50,50,   50));

  stripr.setPixelColor(2, stripr.Color(50,50,   50));

  stripr.setPixelColor(3, stripr.Color(50,50,   50));

  stripl.show();

  delay(200);

  stripr.show();

}

}
```

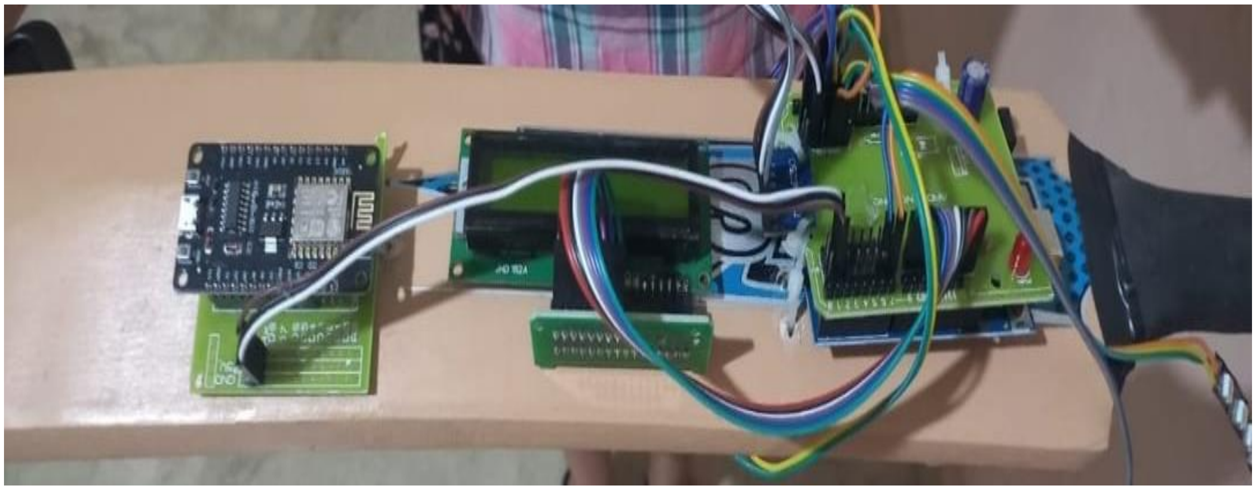
RESULTS AND DISCUSSIONS:

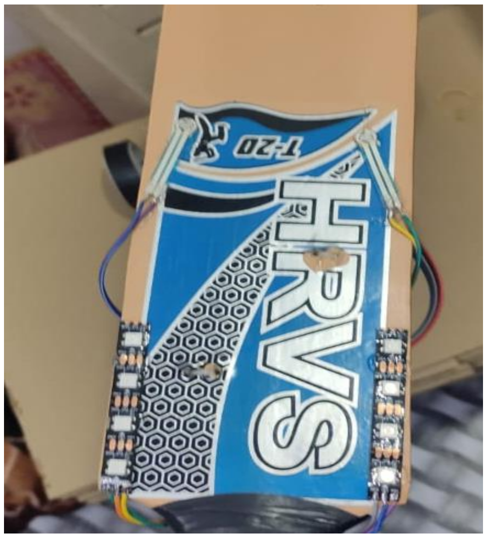
The results of the smart bat implementation were promising, demonstrating its potential to revolutionize player performance analysis in baseball. Through real-time data collection and analysis, the smart bat effectively measured and optimized various swing metrics. Players utilizing the smart bat exhibited improved swing efficiency, with optimized force application and acceleration patterns.

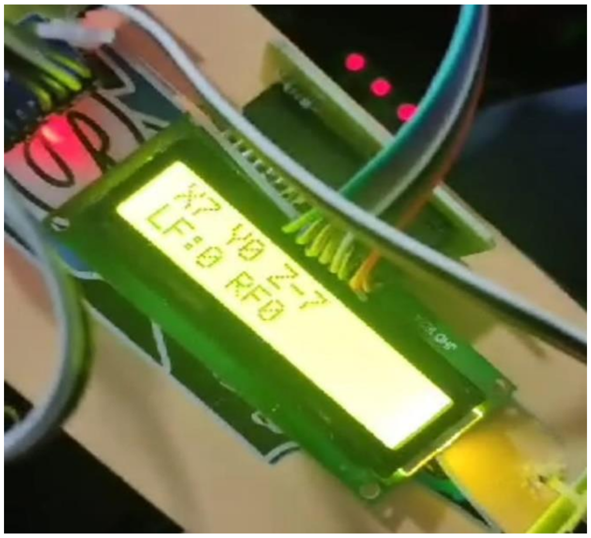
The analysis of swing optimization data indicated a notable enhancement in hitting technique, leading to increased ball contact and more accurate hits. Players showed a significant reduction in energy wastage during swings, translating to improved overall performance. The force and acceleration metrics provided valuable insights into the player's power generation and distribution, enabling targeted training adjustments.

Discussions centered on the smart bat's implications for player development and its integration into coaching strategies. Coaches highlighted the advantage of personalized feedback.

Moreover, the smart bat's ability to track progress over time offered a data-driven approach to skill refinement.







6. CONCLUSION:

In conclusion, the smart bat exhibited substantial potential in optimizing player performance by analyzing a comprehensive range of metrics. Its real-time data insights contributed to enhanced swing efficiency, force application, and acceleration patterns. The implementation of the smart bat opens new avenues for data-driven player development and underscores its role as a valuable tool for modern baseball coaching methodologies.

7. REFERENCES:

- https://www.republicworld.com/technology-news/apps/smart-cricket-bat-sensor-technology-stancebeam-striker.html
- https://morioh.com/a/84695eec26a5/cricket-batting-analysis-with-smart-cricket-bat-sensor
- https://www.researchgate.net/publication/326724905_A_Smart_Bat_Algorithm_for_Wireless_Sensor_Network_Deployment_in_3-D_Environment

—--***-----**