MOTION TRACKING AND ML FOR REHABILITATION

Wrist motion ping-pong game for hand paralyzed pateints

24AIM113 & 24AIM114

TEAM MEMBERS:

SANJAY R -CB.AI.U4AIM24143
SHREEVARSINII B-CB.AI.U4AIM24144
SRIJAN SIVARAM -CB.AI.U4AIM24145
SNENDAR M-CB.AI.U4AIM24127

INTRODUCTION:

- Rehabilitation helps patients recover physical movements after injury or illness.
- Traditional rehab lacks continuous monitoring and feedback.
- Motion tracking using wearable sensors allows real-time monitoring of patient movements.
- Machine learning can analyze these movements and classify them accurately.
- This project combines motion tracking and ML to create a smart, adaptive rehabilitation system.

PROBLEM STATEMENT:

1. **The project uses wrist movements to help with rehabilitation.**
2. **The movements control a game to make therapy more fun and improve recovery.**
3. **Wrist rehabilitation is essential for individuals recovering from wrist fractures (e.g., distal radius fractures), carpal tunnel syndrome, tendon surgery, and sprains.**

OBJECTIVE:

1. **Create a tool for hand-paralyzed patients using wrist movements to create a ping-pong game.**
2. **Track wrist movements (left and right) with the MPU6050 sensor.**
3. **Use LSTM to classify the movements accurately.**
4. **Help patients recover faster by making therapy interactive and engaging.**

LITERATURE REVIEW:

S.No	Paper Title	Author(s)	Key Points
1	A Low-Cost Motion Tracking System for Home-Based Rehabilitation	T. Esfahlani et al. (2016)	Uses IMU sensors for wrist rehab at home, low-cost, and easy setup.
2	Development of a Wrist Rehabilitation Robot	Y. H. Chou et al. (2013)	Designed a robotic system to assist in passive wrist movements.
3	Tele-rehabilitation System for Upper Limb Recovery	A. K. Gupta et al. (2017)	Offers remote rehab through video and sensor feedback.
4	Use of Serious Games in Wrist Rehabilitation	R. J. Mead et al. (2015)	Developed games controlled by wrist movement to increase motivation.
5	Design and Evaluation of a Portable Wrist Exoskeleton	L. M. Maceira-Elvira et al. (2018)	Focused on a wearable device to assist in wrist movement and recovery.

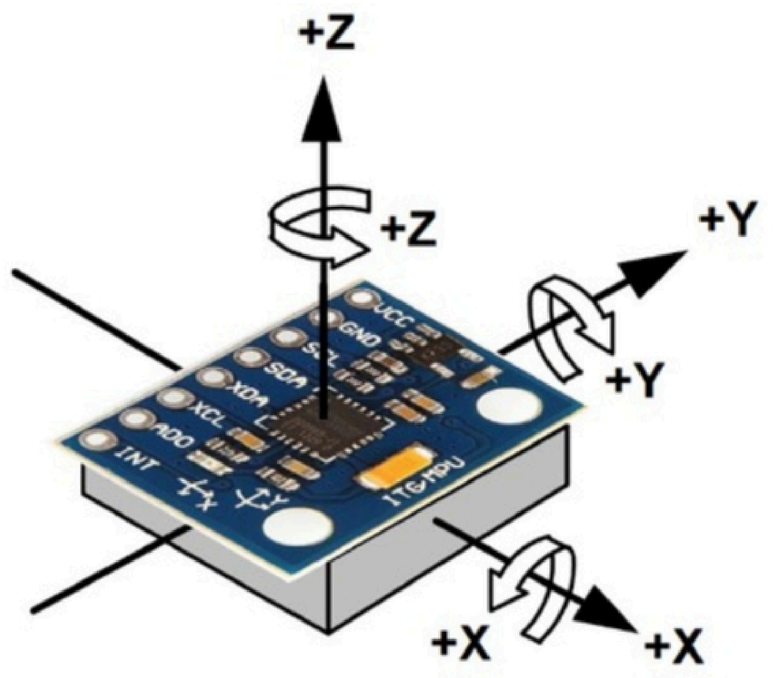
RESEARCH GAPS:

- Not many devices are made for home use.
- Exercises are not customized for each patient.
- People lose interest because rehab is boring.
- Progress is not tracked properly.
- People in villages can’t easily access rehab support.

MPU6050

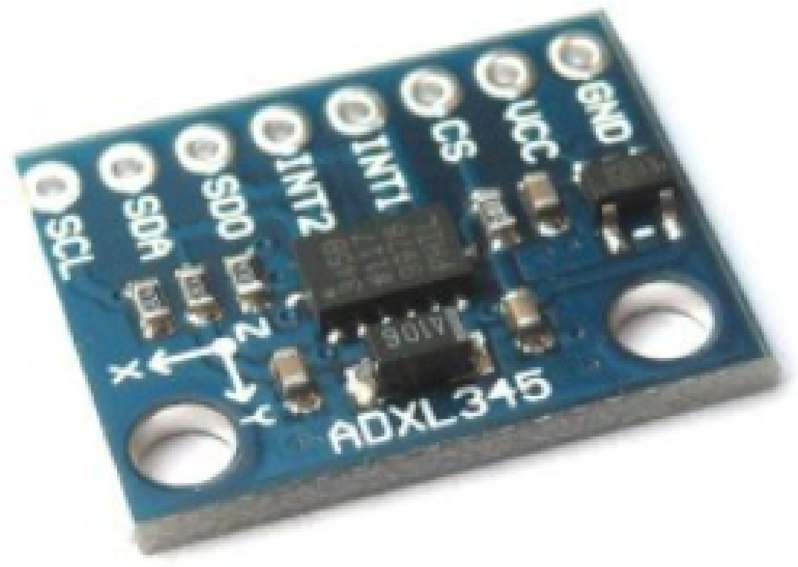
The MPU6050 is a MEMS (Micro-Electro-Mechanical Systems) sensor that combines:

- 3-axis accelerometer to measure acceleration and 3-axis gyroscope to measure angular velocity (rotation).
- It helps in tracking motion and orientation in devices like drones, wearables, and games.
- The sensor communicates with microcontrollers like Arduino via I2C.
- Works with a power supply of 3.3V to 5V.
- MEMS technology allows it to be small, lightweight, and low-power, making it ideal for motion sensing applications.

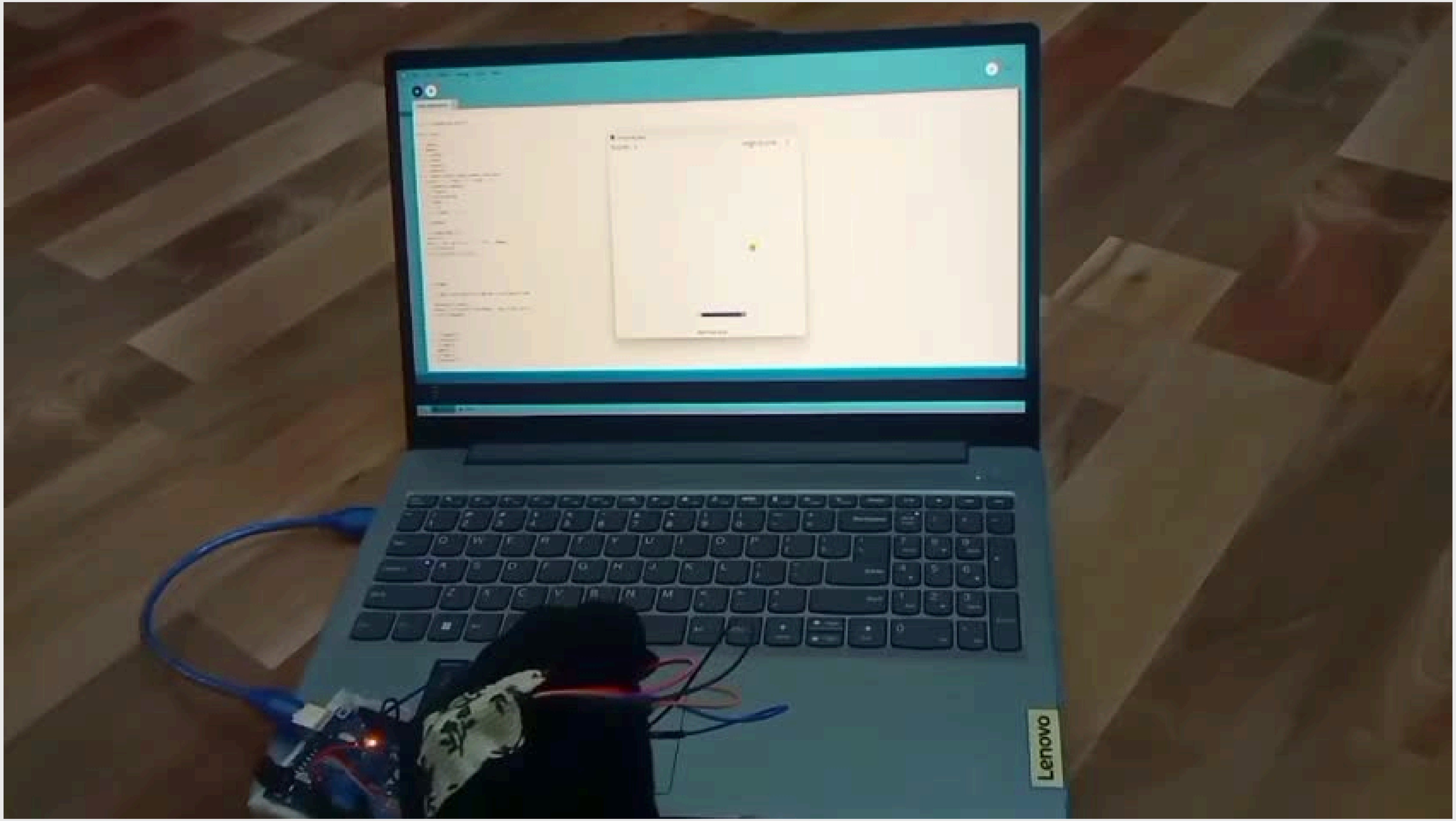


SPECIFICATIONS

Parameter	**Specification**
Operating Voltage (VDC)	3.3V to 5V
Gyroscope Range	±250, ±500, ±1000, ±2000 °/s
Accelerometer Range (g)	±2g, ±4g, ±8g, ±16g
Communication Mode	I2C protocol
Length (mm)	22 mm
Width (mm)	17 mm
Weight (g)	5 g



HARDWARE IMPLEMENTATION



DATA COLLECTION

Raw ax Value Range	Mapped data (0–255)	Action / Movement	Label
ax < -11500	data < 90	Left Hit	2
-11500≤ ax ≤ 11500	90 ≤ data ≤ 160	Idle / No Action	0
ax > 11500	data > 160	Right Hit	1

19	127	0
270	130	0
1856	148	0
2458	154	0
-242	124	0
3735	168	1
592	134	0
1388	142	0
523	133	0
-3966	83	2
-1834	107	0
1490	144	0
3068	161	1
-361	123	0
-28	127	0
3618	167	1
-620	120	0
1476	143	0
3543	166	1
2971	160	0
-824	118	0

collected data rest,left,right

analog_code | Arduino IDE 2.3.4

File Edit Sketch Tools Help

Arduino Uno

analog_code.ino

```cpp

MPU6050 mpu(0x68);

int16_t ax, ay, az;
int16_t gx, gy, gz;

int data, sendData;

void setup()
{
  Wire.begin();
  Serial.begin(9600);
  mpu.initialize();
}

void loop()
{
  sendData = 0;
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
  data = map(ax, -11500, 11500, 0, 255);

  for(int i=0; i<10; i++)
  {
    sendData = sendData + data;
    delay(5);
  }
  sendData = sendData / 10;

  if(sendData >= 255)
      sendData = 255;
  else if(sendData <= 0)
      sendData = 0;

  Serial.println(sendData);
  delay(10);
}
```

Output | Serial Monitor

Message (Enter to send message to 'Arduino Uno' on 'COM5') | New Line | 9600 baud

```
162
162
161
161
163
162
162
163
160
161
```

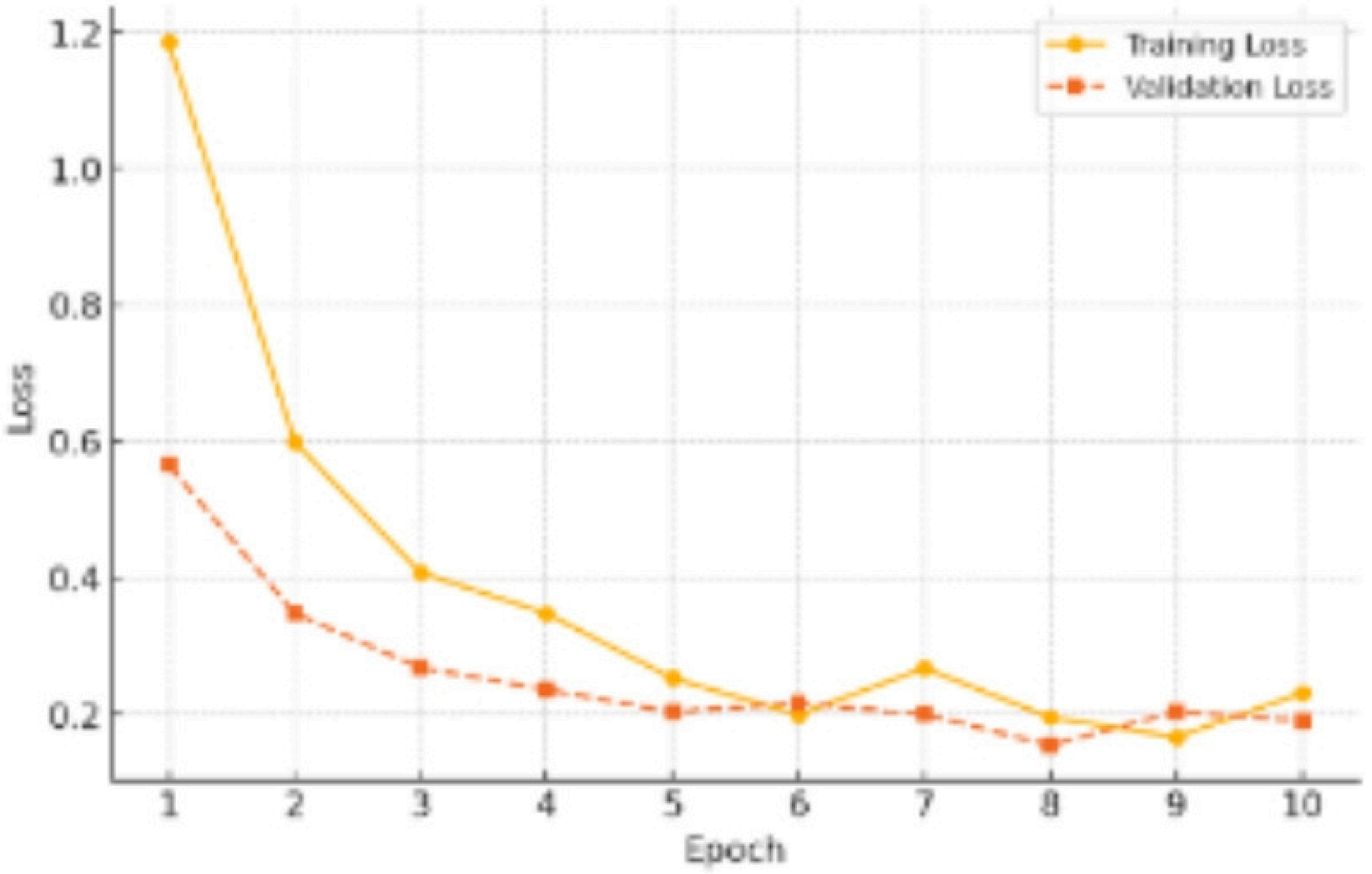
Ln 37, Col 17 Arduino Uno on COM5

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
dataset = pd.read_csv(r"C:\Users\hp\sensor_data_clustered.csv")
features = dataset['Ax'].values
labels = dataset['label'].values  # Labels (assumed to be integer )
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)
def create_sequences(data, labels, seq_length=30):
    sequences = []
    seq_labels = []
    for i in range(len(data) - seq_length):
        sequences.append(data[i:i+seq_length])
        seq_labels.append(labels[i+seq_length])
    return np.array(sequences), np.array(seq_labels)
seq_length = 30
X, y = create_sequences(features_scaled, labels_encoded, seq_length)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = Sequential()
model.add(LSTM(64, input_shape=(X_train.shape[1], X_train.shape[2]), return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(64))
model.add(Dropout(0.2))
model.add(Dense(32, activation='relu'))
model.add(Dense(len(np.unique(y)), activation='softmax'))
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f'Test accuracy: {test_acc}')
model.save(r'C:\Users\hp\sensor_model.h5')
print("Model saved as sensor_model.h5")
```

THE MODEL PERFORMANCE

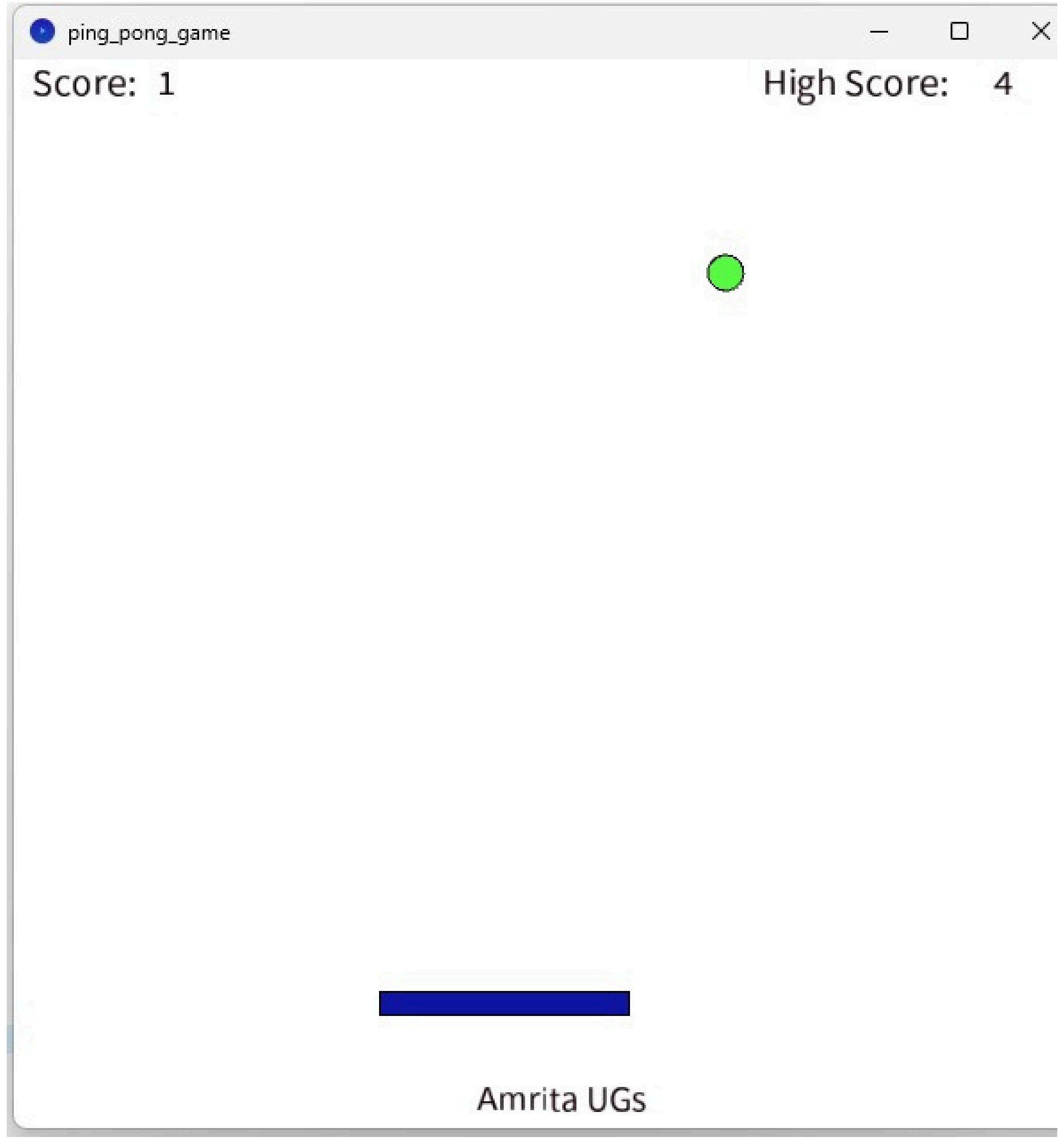
The labeled dataset is involved to perform the LSTM model after the clustering and the model is saved as the h5 file

```
C:\Users\hp\OneDrive - Amrita Vishwa Vidyapeetham\Documents\New folder\Lib\site-packages\keras\src\layers\rnn\rnn.py:200: UserWarning: Do not pass an `i
nput_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instea
d.
  super().__init__(**kwargs)
Epoch 1/10
24/24 ━━━━━━━━━━━━━━━━━━━━ 16s 143ms/step - accuracy: 0.5661 - loss: 1.1854 - val_accuracy: 0.8118 - val_loss: 0.5652
Epoch 2/10
24/24 ━━━━━━━━━━━━━━━━━━━━ 4s 87ms/step - accuracy: 0.8216 - loss: 0.5995 - val_accuracy: 0.8817 - val_loss: 0.3482
Epoch 3/10
24/24 ━━━━━━━━━━━━━━━━━━━━ 2s 78ms/step - accuracy: 0.8586 - loss: 0.4076 - val_accuracy: 0.9247 - val_loss: 0.2811
Epoch 4/10
24/24 ━━━━━━━━━━━━━━━━━━━━ 2s 80ms/step - accuracy: 0.9010 - loss: 0.3253 - val_accuracy: 0.9409 - val_loss: 0.2362
Epoch 5/10
24/24 ━━━━━━━━━━━━━━━━━━━━ 2s 82ms/step - accuracy: 0.9242 - loss: 0.2640 - val_accuracy: 0.9409 - val_loss: 0.2332
Epoch 6/10
24/24 ━━━━━━━━━━━━━━━━━━━━ 2s 72ms/step - accuracy: 0.9358 - loss: 0.1996 - val_accuracy: 0.9462 - val_loss: 0.2163
Epoch 7/10
24/24 ━━━━━━━━━━━━━━━━━━━━ 2s 78ms/step - accuracy: 0.9262 - loss: 0.2677 - val_accuracy: 0.9409 - val_loss: 0.2158
Epoch 8/10
24/24 ━━━━━━━━━━━━━━━━━━━━ 2s 78ms/step - accuracy: 0.9175 - loss: 0.2415 - val_accuracy: 0.9516 - val_loss: 0.2082
Epoch 9/10
24/24 ━━━━━━━━━━━━━━━━━━━━ 2s 78ms/step - accuracy: 0.9340 - loss: 0.1966 - val_accuracy: 0.9355 - val_loss: 0.2105
Epoch 10/10
24/24 ━━━━━━━━━━━━━━━━━━━━ 2s 80ms/step - accuracy: 0.9263 - loss: 0.2312 - val_accuracy: 0.9355 - val_loss: 0.1987
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - accuracy: 0.9354 - loss: 0.1758
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
Test accuracy: 0.9354838728904724
Model saved as sensor_model.h5
```


Training Loss
Validation Loss
1.2
1.0
0.8
0.6
0.4
0.2
Loss
1
2
3
4
5
6
7
8
9
10
Epoch

INTEGRATION OF THE GAME

The model is saved in the h5 file and import to create a simple game named ping-pong game ie the ball that moves as we move the sensor accordingly in the screen



The integration of the ping-pong game

THANK YOU