

IMPLEMENTACIÓN DE MODELOS DE INTELIGENCIA ARTIFICIAL EN UNA TARJETA DE DESARROLLO (Octubre de 2022)

Esteban Ramos Gomez, Diego Ivan Perea, Brayan Castro Balanta

Resumen – En este miniproyecto se realizará una aplicación que se desplegará en una tarjeta de desarrollo como el TinyML Kit de arduino que cuenta con un arduino nano 33 BLE con bluetooth de bajo consumo de energía. Se utilizó este kit de arduino para detectar el movimiento del usuario que realice con el arduino y así mostrar en una aplicación Android de que se trata. En este caso hemos realizado la detección de los movimientos para Tenis, Baloncesto, Golf, Correr y tiro con arco.

Índice de Términos – Deportes, Inteligencia artificial, Arduino, Internet.

I. INTRODUCCIÓN

El uso de microcontroladores para dar solución a problemáticas cotidianas es muy común hoy en día debido a que estos se pueden instalar en cualquier lugar y gracias a que estos microcontroladores trabajan con un consumo energético muy bajo, los convierten en la mejor solución portátil del mercado.

Los wearables inteligentes son una tendencia muy fuerte en el año 2022 debido a que la mayoría poseen un microcontrolador que tiene la capacidad de realizar cálculos complejos en dispositivos de un tamaño muy reducido, también algunos de estos poseen la capacidad de poder procesar pequeñas redes neuronales o algoritmos de clasificación para darle una mayor información al usuario sobre sus rutinas diarias. Para motivos de este proyecto nos inspiramos en los relojes inteligentes para realizar deporte, debido a que estos le proporcionan información adicional al usuario en base a la recolección de datos que realizan por medio de sus sensores, pero en este caso en especial nos enfocaremos en el sensor de acelerómetro que le permite al dispositivo saber si la persona está

corriendo, caminando, haciendo ciclismo, nadando, etc.



Figure 1: Amazfit Bip - GPS Xiaomi

Para la realización de este mini proyecto nos inspiramos en el Amazfit Bip que es un reloj inteligente que tienen diferentes opciones para hacer ejercicio, también incluye un apartado donde la persona con tan solo caminar, correr, trotar, saltar, ir en bicicleta él se dará cuenta de que está realizando el usuario sin que se le indique.

En este caso utilizaremos el microcontrolador Nrf52840 con un voltaje nominal de operación de 3.3V.



Figure 2: Microcontrolador Nrf52840

Este microcontrolador se encuentra ensamblado en la placa de desarrollo Arduino Nano 33 BLE.

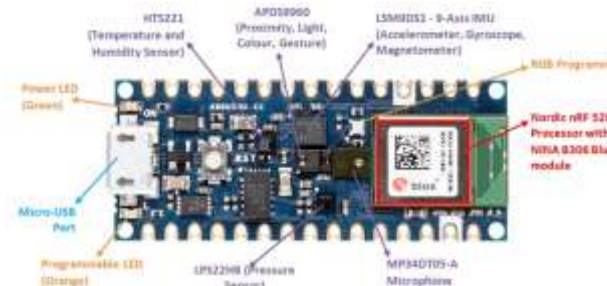


Figure 3: Arduino Nano 33 BLE

El Arduino Nano 33 BLE Sense es una placa completamente nueva en un factor de forma conocido. Viene con una serie de sensores integrados:

- Sensor inercial de 9 ejes: lo que hace que esta placa sea ideal para dispositivos portátiles.
- sensor de humedad y temperatura: para obtener mediciones muy precisas de las condiciones ambientales
- sensor barométrico: podrías hacer una estación meteorológica simple
- micrófono: para capturar y analizar el sonido en tiempo real
- sensor de gestos, proximidad, color de la luz e intensidad de la luz: estima la luminosidad de la habitación, pero también si alguien se está acercando a la pizarra

Con el sensor inercial de 9 ejes haremos la detección de los siguientes deportes:

Tenis: El tenis es un deporte de campo abierto donde las personas hacen uso de los brazos para poder golpear la pelota y este pase al otro lado del campo.

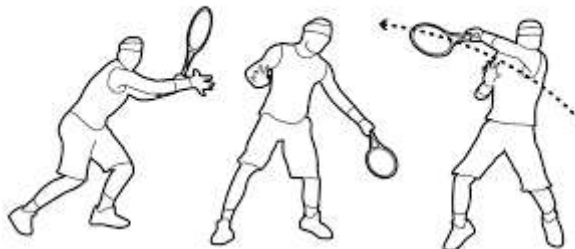


Figure 4: Movimiento de brazos para pegar una pelota en tenis de campo

Baloncesto: El baloncesto es un deporte donde los jugadores deben hacer sestras en la canasta del equipo contrario con ayuda de sus brazos:

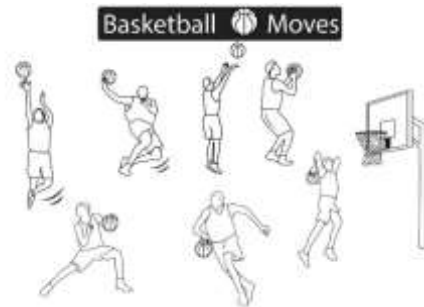


Figure 5: Movimientos para el Baloncesto

Golf: Para jugar golf es necesario hacer un movimiento con los brazos y un palo de golf para impulsar la pelota e introducirla en el hoyo indicado.



Figure 6: Movimientos para jugar golf

Atletismo (Correr): Las personas para correr hacen uso de sus brazos para tomar estabilidad y algo de impulso.

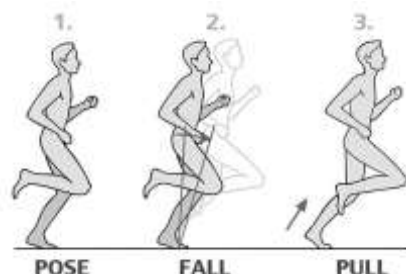


Figure 7: Movimientos para correr

Tiro con arco: Para practicar tiro con arco es necesario hacer uso de los brazos para sostener el arco y estirar la flecha.



Figure 8: Movimientos para tiro con arco

II. PROCEDIMIENTO PARA EL ENVIÓ DEL TRABAJO

A. Etapa de Revisión

Este documento será enviado el 12 de octubre del 2022 por la plataforma de la Universidad Autónoma de Occidente (UAO VIRTUAL), donde nuestro profesor va a realizar las respectivas apreciaciones para nuestra segunda entrega para seminario de ingeniería mecatrónica y así mismo darnos una respectiva calificación, Esperaremos la retroalimentación proporcionada por nuestro profesor para así realizar las correcciones correspondientes para la entrega final de nuestro proyecto

III. INDICACIONES ÚTILES

A. Figuras y tablas



Figure 9: Pines Arduino Nano BLE 33

HC-05 FC-114 & HC-06 FC-114 Connections to Arduino

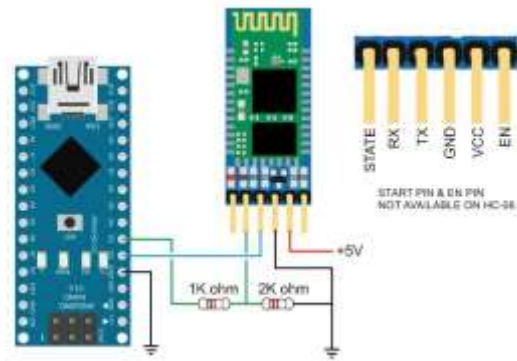


Ilustración 1: Modulo Bluetooth Arduino

IV. ANALISIS Y RESULTADOS

Como se ha mencionado en el resumen y en la introducción, realizaremos una aplicación que tenga la capacidad de distinguir entre si la persona está jugando tenis, baloncesto, golf, practicando tiro con arco o corriendo. Estos deportes es muy el uso de los brazos es esencial e importante para obtener un mejor rendimiento durante una competencia o solamente para realizar ejercicio por lo que es óptimo portar una manilla inteligente, un reloj inteligente o cualquier wearable de manilla para que este registre la actividad del deportista e indicarle que tipo de deporte está realizando.

Edge Impulse es una plataforma para desarrollar algoritmos de aprendizaje maquina enfocados a implementarse en sistemas embebidos como lo son los microcontroladores o computadoras con recursos reducidos. Tiene disponible diversas herramientas que hacen que no debas ser un experto en código para poder implementar tus proyectos.



Ilustración 2: Edge Impulse tomado de: <https://blog.330ohms.com/2021/05/19/que-es-edge-impulse/>

Para empezar con la realización de este mini proyecto, fue necesario tomar 5 muestras para cada uno de los 5 deportes que hemos seleccionado para así asegurarnos de obtener una alta precisión a la hora de entrenar nuestro modelo con la ayuda de Edge Impulse.



Ilustración 3: Análisis Espectral

En este apartado se realiza un análisis espectral a una frecuencia de 100 Hz proporcionada por nuestro acelerómetro, se asignan los ejes X, Y, Z finalizando con el método de clasificación por Keras con NN classifier que nos sirve para procesar y clasificar un conjunto de puntos entre los vecinos más cercanos. Este algoritmo es bastante simple.



Ilustración 4: Selección de parámetros

En la selección de parámetros como nuestro profesor nos indicó, podemos ver la respuesta de los 3 canales que nos ha proporcionado nuestro sensor de acelerómetro. Las señales que tenemos en el apartado izquierdo son señales discretas.



Ilustración 5: Entrenamiento del modelo

Con el paso anterior realizado, seguimos con la etapa de entrenamiento donde ponemos los parámetros de la taza de entrenamiento en 0.0005 los valores de validación en un 20% de los datos totales capturados y finalizamos con un numero de ciclos de formación de 30.

En la parte inferior se nos muestra la arquitectura por defecto de la red neuronal en este caso nos pone las características de la capa de entrada con un valor de 33.



Ilustración 6: Resultados del entrenamiento previo

Como se puede apreciar, los resultados del entrenamiento fueron de alta precisión obteniendo un 95.1% y una pérdida del 0.14. Esto se traduce que los parámetros de entrenamiento, el análisis espectral y finalizando con la toma de datos hecha anteriormente se realizó correctamente por lo que hemos obtenido un modelo con una buena precisión por lo que obtendremos buenos resultados a la hora de implementarlo en nuestro sistema embebido que en este caso es el Arduino nano BLE 33.

Como conclusión para este apartado, se demuestra tiene algunos pequeños problemas con baloncesto y tiro con arco, pero no son valores muy grandes para preocuparnos.



Ilustración 7: Rendimiento del modelo

Se puede apreciar que los puntos rojos en este caso son las predicciones incorrectas que hizo el sistema, este modelo se podría mejorar utilizando más muestras de diferentes deportistas de cada disciplina para así mejorar el rendimiento de la

predicción, ya con estos resultados obtenidos que son buenos procedemos ya a exportar nuestro modelo para nuestro sistema embebido.



Ilustración 8: Exportar modelo Edge Impulse - Arduino



Ilustración 9: Resultado del modelo cuantizado y no optimizado

Ya para finalizar de obtener nuestro modelo por medio de Edge Impulse, se nos da la posibilidad de usar nuestro modelo cuantizado o no optimizado, como se puede apreciar el modelo cuantizado tiene un rendimiento inferior al no optimizado por lo que en este caso se usaremos el modelo no optimizado con una precisión del 74.63%.

Ya con el modelo obtenido por el proceso realizado anteriormente, vamos a la parte del código en arduino para poder cargar nuestro modelo en el sistema embebido mencionado anteriormente.

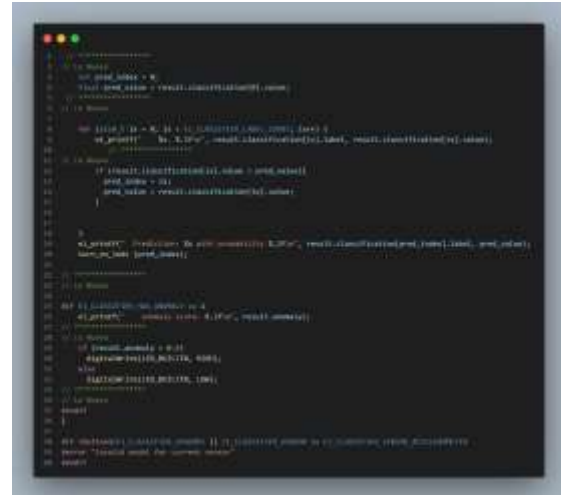


Ilustración 10: Parte 1 del código

Con la función void_puente al momento de dar la clasificación, esta se da con el pred_index que es el index de la clasificación de las clases, por lo que el determinado caso de index activara en HIGH el pin de la clasificación realizada que dio el parámetro pred_index"

```

1  #include <stdio.h>
2
3  int main()
4  {
5      int i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z;
6      char a[256];
7      char b[256];
8      char c[256];
9      char d[256];
10     char e[256];
11     char f[256];
12     char g[256];
13     char h[256];
14     char i[256];
15     char j[256];
16     char k[256];
17     char l[256];
18     char m[256];
19     char n[256];
20     char o[256];
21     char p[256];
22     char q[256];
23     char r[256];
24     char s[256];
25     char t[256];
26     char u[256];
27     char v[256];
28     char w[256];
29     char x[256];
30     char y[256];
31     char z[256];
32     char aa[256];
33     char ab[256];
34     char ac[256];
35     char ad[256];
36     char ae[256];
37     char af[256];
38     char ag[256];
39     char ah[256];
40     char ai[256];
41     char aj[256];
42     char ak[256];
43     char al[256];
44     char am[256];
45     char an[256];
46     char ao[256];
47     char ap[256];
48     char aq[256];
49     char ar[256];
50     char as[256];
51     char at[256];
52     char au[256];
53     char av[256];
54     char aw[256];
55     char ax[256];
56     char ay[256];
57     char az[256];
58     char ba[256];
59     char bb[256];
60     char bc[256];
61     char bd[256];
62     char be[256];
63     char bf[256];
64     char bg[256];
65     char bh[256];
66     char bi[256];
67     char bj[256];
68     char bk[256];
69     char bl[256];
70     char bm[256];
71     char bn[256];
72     char bo[256];
73     char bp[256];
74     char bq[256];
75     char br[256];
76     char bs[256];
77     char bt[256];
78     char bu[256];
79     char bv[256];
80     char bw[256];
81     char bx[256];
82     char by[256];
83     char bz[256];
84     char ca[256];
85     char cb[256];
86     char cc[256];
87     char cd[256];
88     char ce[256];
89     char cf[256];
90     char cg[256];
91     char ch[256];
92     char ci[256];
93     char cj[256];
94     char ck[256];
95     char cl[256];
96     char cm[256];
97     char cn[256];
98     char co[256];
99     char cp[256];
100    char cq[256];
101    char cr[256];
102    char cs[256];
103    char ct[256];
104    char cu[256];
105    char cv[256];
106    char cw[256];
107    char cx[256];
108    char cy[256];
109    char cz[256];
110    char da[256];
111    char db[256];
112    char dc[256];
113    char dd[256];
114    char de[256];
115    char df[256];
116    char dg[256];
117    char dh[256];
118    char di[256];
119    char dj[256];
120    char dk[256];
121    char dl[256];
122    char dm[256];
123    char dn[256];
124    char do[256];
125    char dp[256];
126    char dq[256];
127    char dr[256];
128    char ds[256];
129    char dt[256];
130    char du[256];
131    char dv[256];
132    char dw[256];
133    char dx[256];
134    char dy[256];
135    char dz[256];
136    char ea[256];
137    char eb[256];
138    char ec[256];
139    char ed[256];
140    char ee[256];
141    char ef[256];
142    char eg[256];
143    char eh[256];
144    char ei[256];
145    char ej[256];
146    char ek[256];
147    char el[256];
148    char em[256];
149    char en[256];
150    char eo[256];
151    char ep[256];
152    char eq[256];
153    char er[256];
154    char es[256];
155    char et[256];
156    char eu[256];
157    char ev[256];
158    char ew[256];
159    char ex[256];
160    char ey[256];
161    char ez[256];
162    char fa[256];
163    char fb[256];
164    char fc[256];
165    char fd[256];
166    char fe[256];
167    char ff[256];
168    char fg[256];
169    char fh[256];
170    char fi[256];
171    char fj[256];
172    char fk[256];
173    char fl[256];
174    char fm[256];
175    char fn[256];
176    char fo[256];
177    char fp[256];
178    char fq[256];
179    char fr[256];
180    char fs[256];
181    char ft[256];
182    char fu[256];
183    char fv[256];
184    char fw[256];
185    char fx[256];
186    char fy[256];
187    char fz[256];
188    char ga[256];
189    char gb[256];
190    char gc[256];
191    char gd[256];
192    char ge[256];
193    char gf[256];
194    char gg[256];
195    char gh[256];
196    char gi[256];
197    char gj[256];
198    char gk[256];
199    char gl[256];
200    char gm[256];
201    char gn[256];
202    char go[256];
203    char gp[256];
204    char gq[256];
205    char gr[256];
206    char gs[256];
207    char gt[256];
208    char gu[256];
209    char gv[256];
210    char gw[256];
211    char gx[256];
212    char gy[256];
213    char gz[256];
214    char ha[256];
215    char hb[256];
216    char hc[256];
217    char hd[256];
218    char he[256];
219    char hf[256];
220    char hg[256];
221    char hh[256];
222    char hi[256];
223    char hj[256];
224    char hk[256];
225    char hl[256];
226    char hm[256];
227    char hn[256];
228    char ho[256];
229    char hp[256];
230    char hq[256];
231    char hr[256];
232    char hs[256];
233    char ht[256];
234    char hu[256];
235    char hv[256];
236    char hw[256];
237    char hx[256];
238    char hy[256];
239    char hz[256];
240    char ia[256];
241    char ib[256];
242    char ic[256];
243    char id[256];
244    char ie[256];
245    char if[256];
246    char ig[256];
247    char ih[256];
248    char ii[256];
249    char ij[256];
250    char ik[256];
251    char il[256];
252    char im[256];
253    char in[256];
254    char io[256];
255    char ip[256];
256    char iq[256];
257    char ir[256];
258    char is[256];
259    char it[256];
260    char iu[256];
261    char iv[256];
262    char iw[256];
263    char ix[256];
264    char iy[256];
265    char iz[256];
266    char ja[256];
267    char jb[256];
268    char jc[256];
269    char jd[256];
270    char je[256];
271    char jf[256];
272    char jg[256];
273    char jh[256];
274    char ji[256];
275    char jj[256];
276    char jk[256];
277    char jl[256];
278    char jm[256];
279    char jn[256];
280    char jo[256];
281    char jp[256];
282    char jq[256];
283    char jr[256];
284    char js[256];
285    char jt[256];
286    char ju[256];
287    char jv[256];
288    char jw[256];
289    char jx[256];
290    char jy[256];
291    char jz[256];
292    char ka[256];
293    char kb[256];
294    char kc[256];
295    char kd[256];
296    char ke[256];
297    char kf[256];
298    char kg[256];
299    char kh[256];
300    char ki[256];
301    char kj[256];
302    char kk[256];
303    char kl[256];
304    char km[256];
305    char kn[256];
306    char ko[256];
307    char kp[256];
308    char kq[256];
309    char kr[256];
310    char ks[256];
311    char kt[256];
312    char ku[256];
313    char kv[256];
314    char kw[256];
315    char kx[256];
316    char ky[256];
317    char kz[256];
318    char la[256];
319    char lb[256];
320    char lc[256];
321    char ld[256];
322    char le[256];
323    char lf[256];
324    char lg[256];
325    char lh[256];
326    char li[256];
327    char lj[256];
328    char lk[256];
329    char ll[256];
330    char lm[256];
331    char ln[256];
332    char lo[256];
333    char lp[256];
334    char lq[256];
335    char lr[256];
336    char ls[256];
337    char lt[256];
338    char lu[256];
339    char lv[256];
340    char lw[256];
341    char lx[256];
342    char ly[256];
343    char lz[256];
344    char ma[256];
345    char mb[256];
346    char mc[256];
347    char md[256];
348    char
```

```

1 // =====
2 // 1.1.1.1.1.1.1
3 // 1.1.1.1.1.1.1
4 // 1.1.1.1.1.1.1
5 // 1.1.1.1.1.1.1
6 // 1.1.1.1.1.1.1
7 // 1.1.1.1.1.1.1
8 // 1.1.1.1.1.1.1
9 // 1.1.1.1.1.1.1
10 // 1.1.1.1.1.1.1
11 // 1.1.1.1.1.1.1
12 // 1.1.1.1.1.1.1
13 // 1.1.1.1.1.1.1
14 // 1.1.1.1.1.1.1
15 // 1.1.1.1.1.1.1
16 // 1.1.1.1.1.1.1
17 // 1.1.1.1.1.1.1
18 // 1.1.1.1.1.1.1
19 // 1.1.1.1.1.1.1
20 // 1.1.1.1.1.1.1
21 // 1.1.1.1.1.1.1
22 // 1.1.1.1.1.1.1
23 // 1.1.1.1.1.1.1
24 // 1.1.1.1.1.1.1
25 // 1.1.1.1.1.1.1
26 // 1.1.1.1.1.1.1
27 // 1.1.1.1.1.1.1
28 // 1.1.1.1.1.1.1
29 // 1.1.1.1.1.1.1
30 // 1.1.1.1.1.1.1
31 // 1.1.1.1.1.1.1
32 // 1.1.1.1.1.1.1
33 // 1.1.1.1.1.1.1
34 // 1.1.1.1.1.1.1
35 // 1.1.1.1.1.1.1
36 // 1.1.1.1.1.1.1
37 // 1.1.1.1.1.1.1
38 // 1.1.1.1.1.1.1
39 // 1.1.1.1.1.1.1
40 // 1.1.1.1.1.1.1
41 // 1.1.1.1.1.1.1
42 // 1.1.1.1.1.1.1
43 // 1.1.1.1.1.1.1
44 // 1.1.1.1.1.1.1
45 // 1.1.1.1.1.1.1
46 // 1.1.1.1.1.1.1
47 // 1.1.1.1.1.1.1
48 // 1.1.1.1.1.1.1
49 // 1.1.1.1.1.1.1
50 // 1.1.1.1.1.1.1
51 // 1.1.1.1.1.1.1
52 // 1.1.1.1.1.1.1
53 // 1.1.1.1.1.1.1
54 // 1.1.1.1.1.1.1
55 // 1.1.1.1.1.1.1
56 // 1.1.1.1.1.1.1
57 // 1.1.1.1.1.1.1
58 // 1.1.1.1.1.1.1
59 // 1.1.1.1.1.1.1
60 // 1.1.1.1.1.1.1
61 // 1.1.1.1.1.1.1
62 // 1.1.1.1.1.1.1
63 // 1.1.1.1.1.1.1
64 // 1.1.1.1.1.1.1
65 // 1.1.1.1.1.1.1
66 // 1.1.1.1.1.1.1
67 // 1.1.1.1.1.1.1
68 // 1.1.1.1.1.1.1
69 // 1.1.1.1.1.1.1
70 // 1.1.1.1.1.1.1
71 // 1.1.1.1.1.1.1
72 // 1.1.1.1.1.1.1
73 // 1.1.1.1.1.1.1
74 // 1.1.1.1.1.1.1
75 // 1.1.1.1.1.1.1
76 // 1.1.1.1.1.1.1
77 // 1.1.1.1.1.1.1
78 // 1.1.1.1.1.1.1
79 // 1.1.1.1.1.1.1
80 // 1.1.1.1.1.1.1
81 // 1.1.1.1.1.1.1
82 // 1.1.1.1.1.1.1
83 // 1.1.1.1.1.1.1
84 // 1.1.1.1.1.1.1
85 // 1.1.1.1.1.1.1
86 // 1.1.1.1.1.1.1
87 // 1.1.1.1.1.1.1
88 // 1.1.1.1.1.1.1
89 // 1.1.1.1.1.1.1
90 // 1.1.1.1.1.1.1
91 // 1.1.1.1.1.1.1
92 // 1.1.1.1.1.1.1
93 // 1.1.1.1.1.1.1
94 // 1.1.1.1.1.1.1
95 // 1.1.1.1.1.1.1
96 // 1.1.1.1.1.1.1
97 // 1.1.1.1.1.1.1
98 // 1.1.1.1.1.1.1
99 // 1.1.1.1.1.1.1
100 // 1.1.1.1.1.1.1

```

Llamada de métodos de clasificación para imprimirlos en serial la clasificación realizada y porcentaje de precisión de clasificación.

Para la implementación física hemos usando un arduino uno auxiliar para establecer la conexión mediante bluetooth por medio de un módulo HC-05 de con el siguiente esquema:

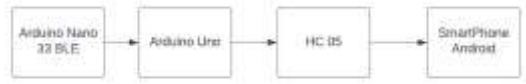


Ilustración 14: Esquema de proceso

Como se puede apreciar en el esquema de proceso, vamos a poner nuestro Arduino Nano 33 BLE como controlador principal debido a que este tiene el modelo de inteligencia artificial que hemos creado a partir de Edge Impulse, seguidamente esta ira a un arduino uno que será el controlador de la conexión serial por medio del HC-05 a nuestro SmartPhone con Android donde se va a visualizar que deporte está practicando la persona.

Ahora procedemos a mostrar por medio de imágenes lo que hemos utilizado para poder realizar este mini proyecto real.

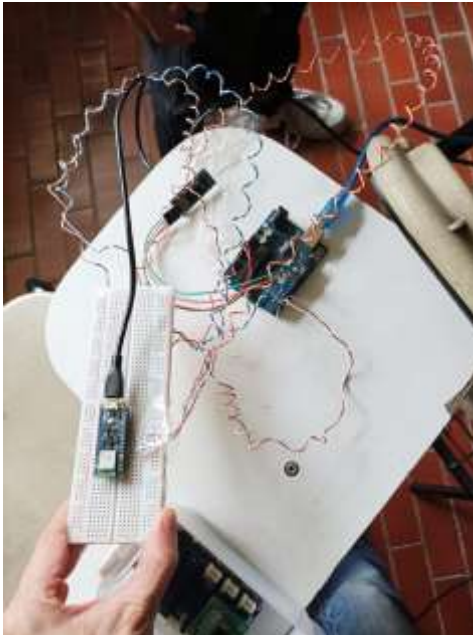
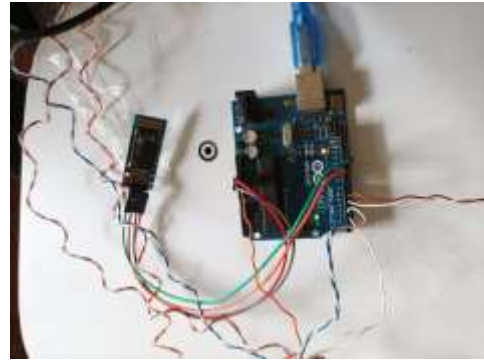


Ilustración 15: Conexion Arduino Nano BLE 33 con arduino Uno + Modulo Blueooth



Con estas conexiones hemos conectado el arduino Nano 33 BLE y el arduino uno para que este pueda enviar por medio del módulo blueooth de arduino HC-05 a nuestro celular Android para mostrar el deporte que está realizando el usuario.



Ilustración 16: Pruebas de movimiento de las manos para la predicción del deporte realizado

Nuestro compañero tomo el Arduino Nano 33 BLE para probar el modelo entrenado en Edge Impulse.



Ilustración 17: Visualización de la predicción del modelo por medio de a app creada en Android Studio

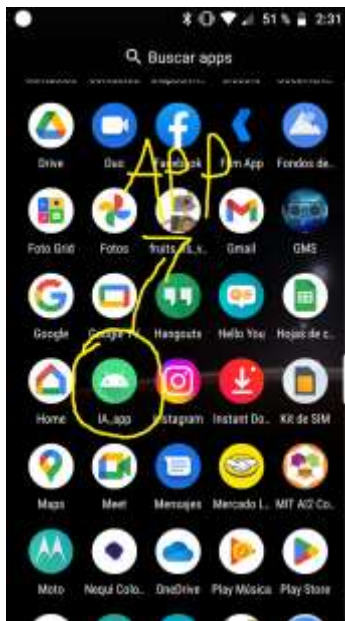


Ilustración 18: App diseñada en Android Studio



Ilustración 19: Visualización de la App

IX. CONCLUSIÓN

- La inteligencia artificial aplicada a detección de movimientos utilizados con el acelerómetro nos da la precisión y comportamientos adecuadas para controlar diferentes sistemas, tener ia en sistemas embebidos amplía la gama de funcionalidades que se efectúan en la vida cotidiana y mejora el desempeño de esta y como estas se pueden medir.
 - El uso de Android para aplicaciones de inteligencia artificial es muy interactivo debido a que tiene una gran comunidad detrás.
-
- IX. REFERENCIAS
- [1] "¿Qué es la Inteligencia Artificial? - Iberdrola". Iberdrola. <https://www.iberdrola.com/innovacion/que-es-inteligencia-artificial> (accedido el 12 de octubre de 2022).
- [2] "Nano 33 BLE Sense | Arduino Documentation | Arduino Documentation". Arduino Docs | Arduino Documentation | Arduino Documentation. <https://docs.arduino.cc/hardware/nano-33-ble-sense> (accedido el 12 de octubre de 2022).
- [3] "Como empezar con Arduino Nano 33". kolwidi. <https://kolwidi.com/blogs/blog-kolwidi/como-empezar-con-arduino-nano-33> (accedido el 12 de octubre de 2022).
- [4] "How to use Edge Impulse to train machine learning models for Raspberry Pico - Arducam". Arducam. <https://www.arducam.com/docs/pico/arducam-pico4mltinymldewkit/how-to-use-edge-impulse-to-train-machine-learning-models-for-raspberry-pico/> (accedido el 12 de octubre de 2022).
- Es de suma importancia tener en cuenta la posición inicial del Arduino Nano BLE 33 a la hora de tomar los datos de movimiento debido a que en esta práctica del mini proyecto 2.
 - Tuvimos muchos problemas a poder utilizar el Bluetooth del Arduino debido a que no se encontraba la información necesaria o precisa sobre el tema.