futuro (muy caro)

Sensores: Flex, MPU 6050 | Adicionales: TCA9548A

Actuadores:

Microcontrolador: ESP32

Software: Unity

- Cosas a manejar/saber:

-Quaternion (o no)

-Conexión µcontrolador - Unity ~

-Gasto energía

- Dimensiones y pesos

MPU 6050: 21,2 mm x 16,4 mm x 3,3 mm 3,1g

ESP32: 48 x 26 x 11,5 10g

- Conexión S-H: Se programa en C en Unity :), se conecta mediante puerto serial (se puede con bluetooth solo es saber el COM y configurar)

Valores en escalas de metros

Consola solo lee los print ln (o una fila de datos print que termine en println)

Solo entiende 1 script

Agregar una f a un número se hace que sea tipo float

transform.position

transform.rotation o transform.rotate → rota siempre lo que le diga

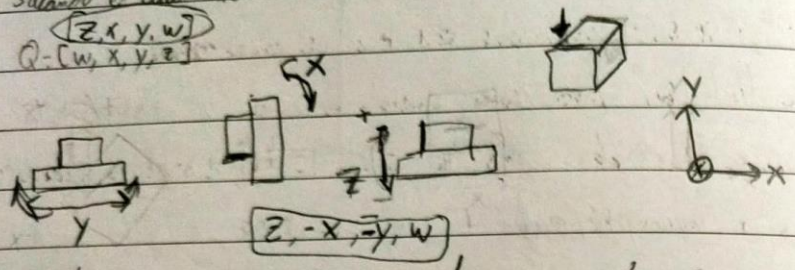
Quaternion.Euler(float x, float y, float z)

↳ lo deja quieto al ángulo que le dice

COM 7 → entrante COM 8 → Saliente

Sacando el cuaternion es directo usar el transform.rotation

Q-[w, x, y, z]
[z, x, y, w]



[z, -x, -y, w]

Cuaternion → no puede representar una rotación que exceda 180°

Para que no salga el error de I.O es que todo debe salir del mismo script y mismo objeto

el COM que sea saliente

el cuaternión va de [1, -1] ya que está compuesto por cos y sin,

Un cuaternión se escribe así: $\cos(\theta/2) + i\sin(\theta/2) + j\sin(\theta/2) + k\sin(\theta/2)$

$w + i(x) + j(y) + k(z)$

donde θ es el angulo de rotación

MPU-6050

-3 ADC de 16 bits para cada salida de los giroscopios

-3 ADC de 16 bits para cada salida de los acelerómetros

giroscopio programable rango de ±250, ±500, ±1000 y ±2000°/seg

acelerómetro programable rango de ±2g, ±4g, ±8g y ±16g (g = 9,8 m/s²)

VDD: 2,375V - 3,46V [Vlogic: 1,71V - VDD]

Comunicación I^2C

Operating current (todo + DMP on): 3,9mA

Giroscopio: 3,6 mA - standby: 5μA

Acelerometro: 500μA - Low power: 10μA 1,25Hz, 20μA 5Hz, 60μA 20Hz, 110μA 40Hz

Digital Motion Processing (DMP) interno, admite algoritmos de reconocimiento de gestos y procesamiento de movimiento 3D (*)

I^2C direcciones: ADO=0 1101000, ADO=1 1101001

ESP32

De todos, mejor probar

Pines ADC: ~~D2, D4,~~ D12, D13, D14, D15, ~~D25, D26, D27, D32, D33, D34, D35~~

↓ Tienen que estar desconectados al hacer el upload

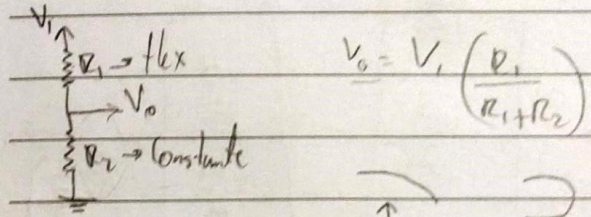
Pines PWM: ~~16, 17, 18, 19, 21, 22, 23, 25, 26, 27, 32, 33~~

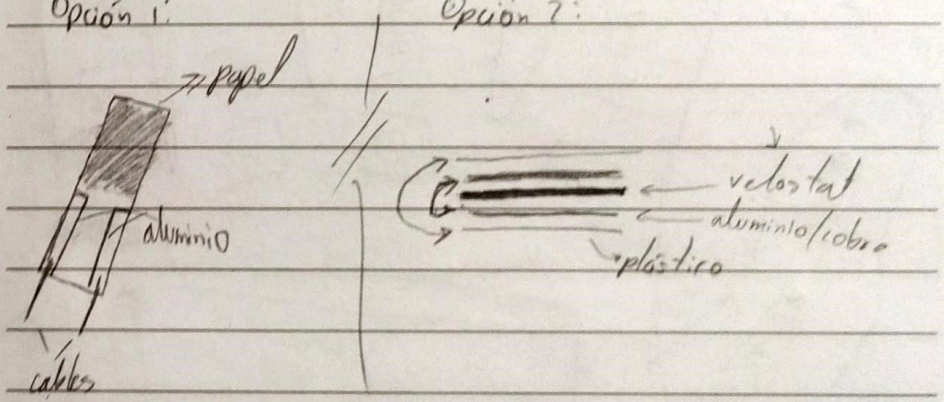
Cambiar: 25-27, 3233 ⇒ D2, D14, D15. Puedo cambiar D4?

16 → RX2 17 → TX2

Sensor flex casero

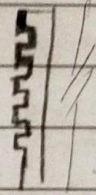
50 3,3?





Será necesario calibrar cada 1, en total son 10

o imprimir

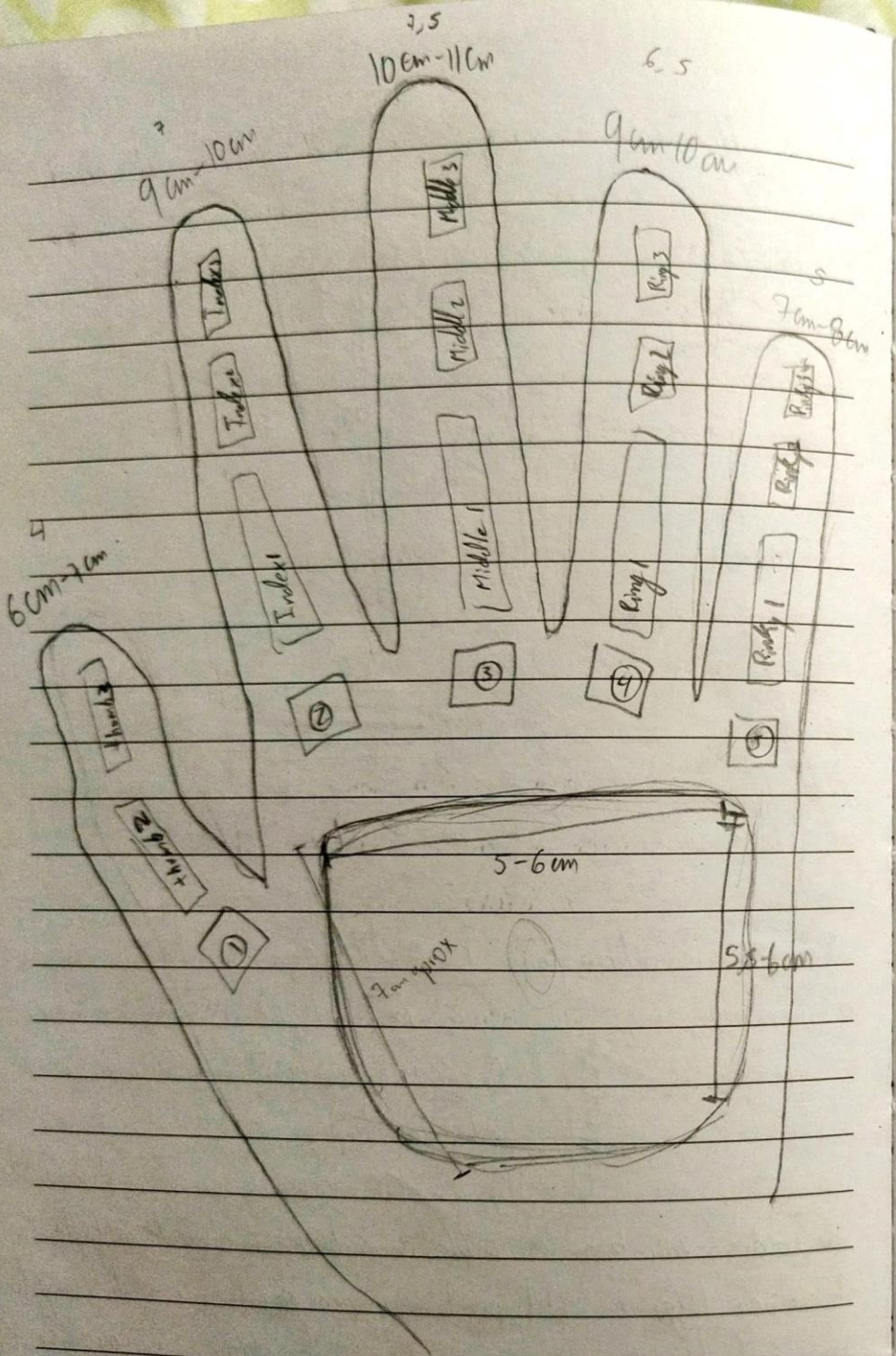


Opción 1: Con lápiz parece que es mejor

Opción 2: Parece que lo mejor sería en la mitad con lápiz

Hay que crear sensores con menos probabilidades de ruidos, unir bien los cables, crear la versión con cinta aislante

la versión larga es útil por lo menos mejor para lo que es el dedo completo pero nada fijo. Es mejor la opción de que estén sobre ya que hay mejor variación. La versión corta hacerla más larga. Hacer uno del que sea mitad/mitad



Papel siempre dobles, parece que entre más delgado mejor, el centímetro centímetro y medio, cubrir el ancho del aluminio y que tan largo van 7 intentos solo 1 ha salido bien y con cinta aislante

④→✓ ⑤→✓ ②→✓

DIS→ADC funciona

Izq↑: 4, 3, 5, 6, 7, 6, 10, 9, 12, 11 (3-12) → pines ADC

Pared: 3

①→D25 ②→D33 ③→D32 ④→D27 ⑤→D26

4135 → los 5 a 3,3V

⑤→flex3 3900~4000 un poquito
3700-3800 mitad
3300-3400 completo
$-869,5\ln(x)+7238,1$
off: 9,1

③→flex4 4100 un poquito
3900-4000 mitad
3600-3750 completo
$-1604\ln(x)+13366$
off: -18,91 -4,86

②→flex5 4050-4100 un poquito
3900-4000 medio
3600-3700 completo
$-1360\ln(x)+11339$
off: -2,44

④→flex2 2600-3000 un poquito
2000-2100 mitad
1500-1600 completo
$-179,7\ln(x)+1477,5$
off: 13,99

Ancho aluminio: 0,3 ; Ancho sensor: 1 cm → Se quiere todos así

Largo: Depende del dedo:

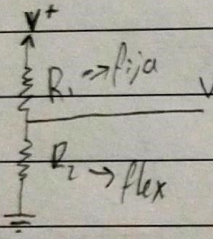
Nuevos pines:

D27 → (3) D26 → (4) D25 → (5) D33 → (1) D32 → (2)

Nuevos nuevos con colores:

D33 → (1) D32 → (2) D25 → (3) D27 → (4) D26 → (5)

↓ BA corto ↓ BC corto ↓ BC largo ↓ BV ↓ BA largo



R_1 se puede cambiar a valores de MΩ en caso de que no sirva con 10K para que sea más sensible

Probar a hacer todos los sensores multifilares y con la cinta de aluminio, de paso agregar un punto de soldadura si no parecen agarrarse bien

164 datos (durante 30 segundos cada 150ms) → desviación estándar

UNITY

La lógica para programarlo es que de un script general se obtiene todo y se hacen métodos públicos a los cuales después acceder en los scripts que correspondan a cada objeto que se va a mover. Para eso hay que acceder al objeto y en este caso además acceder al hijo del objeto porque ahí es donde está guardado el script general

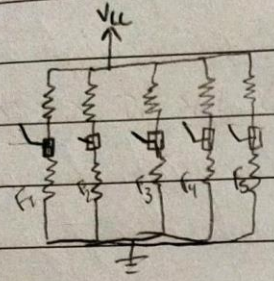
5x7 cm

24x18

8

38 - 38

1,9x18,91



ring palm collider base1 marker — ColliderPalm

index1 — ColliderIndex1

middle1 — ColliderMiddle1

pinky1 — ColliderPinky1

Ring1 — ColliderRing1

Thumb2 — ColliderThumb2

BUZZERS

1→18, 2→19, 3→16, 4→17, 5→23, 6→25, 7→26

Pasar a orden:

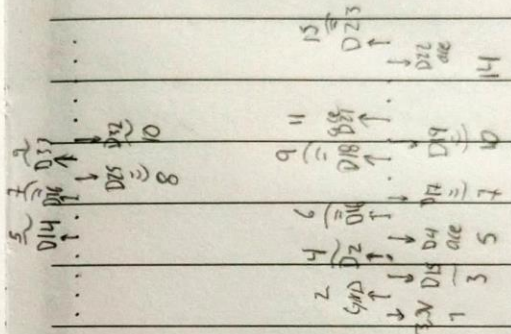
1→23, 2→19, 3→18, 4→17, 5→16, 6→25, 7→26

Pines nuevos nuevos:

Flex: D2 → (3), D14 → (2), D15 → (1), D33 → (4), D32 → (5)

Buzzers: D23 → (6), D19 → (4), D18 → (3), D17 → (2), D16 → (5), D25 → (7)

D26 → (1) Acelerometro:



BA corto → D14, BC corto → D26, BC bajo → D25, BV → D33, BM bajo → D32

0 → 17 → Palma

1 → 18 → Pinky

2 → 19 → Ring

3 → 23 → Thumb

4 → 25 → Index

5 → 26 → Middle