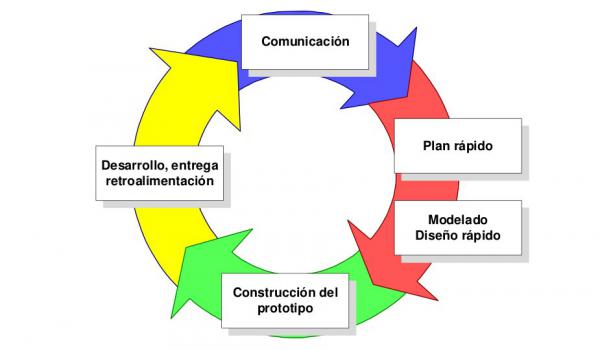
CAPITULO 3 METODOLOGIA DE DESARROLLO

3.1 Metodología prototipo

El Modelo de prototipos, en Ingeniería de software, pertenece a los modelos de desarrollo evolutivo. El prototipo debe ser construido en poco tiempo, usando los programas adecuados y no se debe utilizar muchos recursos.

El diseño rápido se centra en una representación de aquellos aspectos del software que serán visibles para el cliente o el usuario final. Este diseño conduce a la construcción de un prototipo, el cual es evaluado por el cliente para una retroalimentación; gracias a ésta se refinan los requisitos del software que se desarrollará. La interacción ocurre cuando el prototipo se ajusta para satisfacer las necesidades del cliente. Esto permite que al mismo tiempo el desarrollador entienda mejor lo que se debe hacer y el cliente vea resultados a corto plazo.

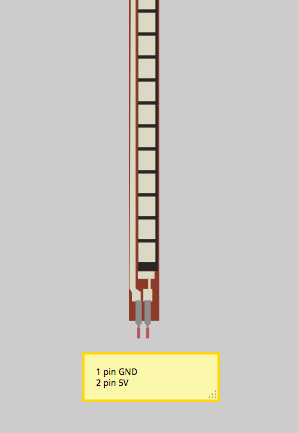


Prototipo 1 Conexión física con el Arduino y el flexómetro

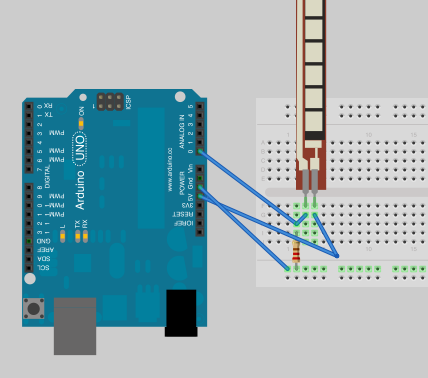
En este primer prototipo realizaremos la conexión de Arduino con el flexómetro para poder detectar las señales que emitirá.

Ahora hablemos primero del flex sensor para saber que estamos usando, dicho sensor tiene en sí varios sensores que cambian en la resistencia en función de la cantidad de curvatura que se le emplea, convierten el cambio en la curva a la resistencia eléctrica a mayor sea la curva, mayor será el valor de la resistencia

Su principal desarrolló inicial fue para las bolsas de aire para los carros, hoy en día se usa hasta en bocinas de los carros, juguetes que detectan diferentes grados de presión o de flexión entre otros, en nuestro caso será utilizada para el control sobre los objetos modelados.  
  
Ahora veamos los pines del flex sensor:

[](http://4.bp.blogspot.com/-PAtoZyZSG6E/T4h11yf_7oI/AAAAAAAAA8Q/9ZhwHW0ZRZ0/s1600/Captura+de+pantalla+2012-04-13+a+la%28s%29+13.46.31.png)

Ahora que entendemos que es, necesitaremos una resistencia de 330 ohms y el flex sensor.Una vez obtenido el material pasaremos al circuito.

[](http://4.bp.blogspot.com/-6l3SPEG3PWk/T4h19QLYbaI/AAAAAAAAA8Y/3W4aE5tBwNM/s1600/Captura+de+pantalla+2012-04-13+a+la%28s%29+13.45.11.png)

3.3 Prototipo 2 Conexión con Arduino nano al guante

Se emplean en los guantes para detectar el movimiento del dedo, para controlar el movimiento sobre los objetos modelados en Blender.

[](http://4.bp.blogspot.com/-Bkue0Wb6zwc/T4h1nkOyBvI/AAAAAAAAA8I/gb8bKT2k_XQ/s1600/hypersenseglove.jpg)

3.4 Prototipo 3 Conexión Arduino a Python

INSTALAR PYTHON Y PYSERIAL

Lo primero que necesitamos es tener instalado Python en nuestro dispositivo. Si aún no te has iniciado con Python puedes consultar la entrada [Nuestro primer programa en Python](https://www.luisllamas.es/primer-programa-python/) donde vimos cómo instalar Python en Windows y Linux, y unos ejemplos básicos para introducir su uso.

Una vez que tengamos Python instalado para poder comunicarnos con Arduino necesitamos la librería PySerial, que nos permite emplear de forma sencilla el puerto serie. La librería PySerial está disponible en este enlace <https://github.com/pyserial/pyserial>

Descargamos y ejecutamos el instalador, para añadir la librería PySerial a nuestra instalación de Python.

También podemos instalar la librería PySerial directamente desde Python, escribiendo el siguiente comando desde una consola.

|  |  |
| --- | --- |
| 1 | python -m pip install PySerial |

Con cualquiera de los dos métodos, al final tendremos la librería PySerial instalada y lista para ser utilizada en nuestros proyectos.

### RECIBIR INFORMACIÓN DESDE ARDUINO

En este primer ejemplo vamos a leer información enviada por Arduino, y capturada y mostrada en pantalla por Python.

Para ello empezamos cargamos en Arduino el siguiente sketch, que simplemente envía continuamente una vez por segundo el texto “Hola Mundo”.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | void setup() {  Serial.begin(9600);  }    void loop() {  Serial.println("Hola mundo");  delay(1000);  } |

Dejamos el sketch funcionando en Arduino, y vamos a realizar el script en Python. Creamos un nuevo archivo de texto vacio, que guardamos con el nombre "read.py". En su interior copiamos el siguiente código.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | import serial, time  arduino = serial.Serial('COM4', 9600)  time.sleep(2)  rawString = arduino.readline()  print(rawString)  arduino.close() |

Lo que hacemos es importar la librería Serial (PySerial) e instanciar un objeto PySerial, que hemos llamado "arduino". En el constructor del objeto Serial pasamos los parámetros del puerto serie que estemos empleado.

Recuerda sustituir el puerto serie del código, en el ejemplo "COM4", por el puerto serie en el que tengas conectado Arduino.

A continuación, empleamos la orden “readline()” del objeto Serial para leer una línea enviada por Arduino. Mostramos la línea en pantalla mediante la orden “Print()”

Finalmente mediante la orden “close()" cerramos el puerto serie.

Como vemos, emplear el puerto serie con PySerial es realmente sencillo. Lo único que puede parecer extraño es por qué hemos tenido que importar la librería “time”.

El motivo es que desde que creamos el objeto Serial hasta realmente está disponible para ser usado, se necesita un cierto tiempo para abrir el puerto serie. Por tanto, tenemos que introducir una espera mediante la función “Sleep”, que pertenece a la librería “time”.

### ENVIAR INFORMACIÓN A ARDUINO

En este segundo ejemplo vamos a enviar datos a Arduino desde Python. Para ello vamos a usar el siguiente sketch que vimos en la entrada [Comunicación de Arduino con puerto serie.](https://www.luisllamas.es/arduino-puerto-serie/)

Este sketch recibe un número desde 1 a 9 y hace parpadear el LED integrado, conectado al PIN13, el número de veces recibido. Cargamos el sketch en Arduino, e igual que antes lo dejamos funcionado.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | const int pinLED = 13;    void setup()  {     Serial.begin(9600);     pinMode(pinLED, OUTPUT);  }    void loop()  {     if (Serial.available()>0)     {        char option = Serial.read();        if (option >= '1' && option <= '9')        {           option -= '0';           for (int i = 0;i<option;i++)           {              digitalWrite(pinLED, HIGH);              delay(100);              digitalWrite(pinLED, LOW);              delay(200);           }        }     }  } |

En la parte de Python necesitaremos un nuevo script que llamaremos, por ejemplo "write.py". En su interior pegamos el siguiente código.

|  |  |
| --- | --- |
| 1  2  3  4  5 | import serial, time  arduino = serial.Serial("COM4", 9600)  time.sleep(2)  arduino.write(b'9')  arduino.close() |

Como vemos la escritura es muy similar a la lectura. En primer lugar importamos la librería PySerial y creamos un nuevo objeto de tipo Serial, indicando los valores del puerto serie que estemos empleando.

En esta ocasión, **escribimos el valor mediante la función “write”** (en el ejemplo, 9). La función “Write” envía bytes, por lo que **es necesario convertir el valor a bytes** antecediendo una b al valor enviado (en el ejemplo b’9’).

Por último, cerramos el puerto serie con la función "close()".

Nuevamente hemos tenido que importar la librería time, para poder usar la función Sleep, y dar un tiempo entre el inicio de la conexión del puerto serie y el envío de datos.

Con esto ya tenemos las **funciones básicas para enviar y recibir información a Arduino desde Python**, empleando la librería PySerial para controlar el puerto serie. Resulta sencillo integrar estas funciones en nuestros programas.

3.5 Prototipo 4 Manipulación de datos de Python hacia Blender

Una vez que tengamos Python instalado para poder comunicarnos con Arduino necesitamos la librería PySerial, que nos permite emplear de forma sencilla el puerto serie. La librería PySerial está disponible en este enlace <https://github.com/pyserial/pyserial>

Descargamos y ejecutamos el instalador, para añadir la librería PySerial a nuestra instalación de Python.

También podemos instalar la librería PySerial directamente desde Python, escribiendo el siguiente comando desde una consola.

|  |  |
| --- | --- |
| 1 | python -m pip install PySerial |

Con cualquiera de los dos métodos, al final tendremos la librería PySerial instalada y lista para ser utilizada en nuestros proyectos.

Con esto ya tenemos las **funciones básicas para enviar y recibir información a Arduino desde Python**, empleando la librería PySerial para controlar el puerto serie. Resulta sencillo integrar estas funciones en nuestros programas.

3.6 Prototipo 5 Implementación y resultados

Aquí se muestra la implemtentacion de cada prototipo en uno que siendo el ultimo ya que sera el que podrá manipular los objetos modelados en Blender.