Instituto Tecnológico de Buenos Aires

31.99 Mecatrónica Aplicada

# Trabajo Práctico: Sensor Fusion

Parra, Rocío 57669

*Profesores*

Perfumo, Lucas Alberto
Fortunatti, Nelson Ariel

Presentado: 23/10/2020

# Índice



# 1. Consigna

- Diagrama de bloques de los dos programas
- Sincronización
- Presentar gráficos obtenidos
- Video

# 2. Funcionamiento

## 2.1. Programa del microcontrolador

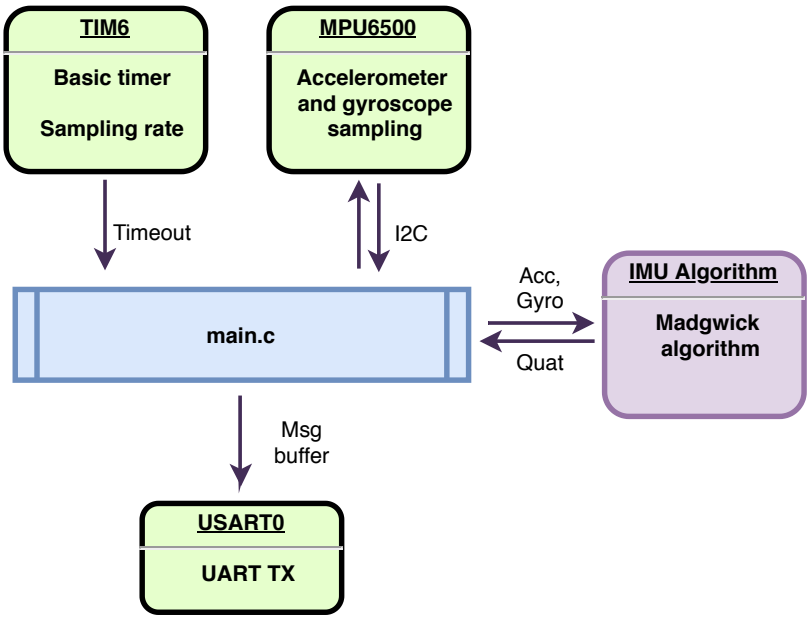


Figura 1: Diagrama de bloques del programa del microcontrolador

## 2.2. Programa de la computadora

# 3. Sincronización

El envío de datos desde el microprocesador a la computadora se realizó vía UART, con un baudrate de 115200, sin paridad, 8 bits por palabra y un stopbit. No se implementó ningún mecanismo de acknow-

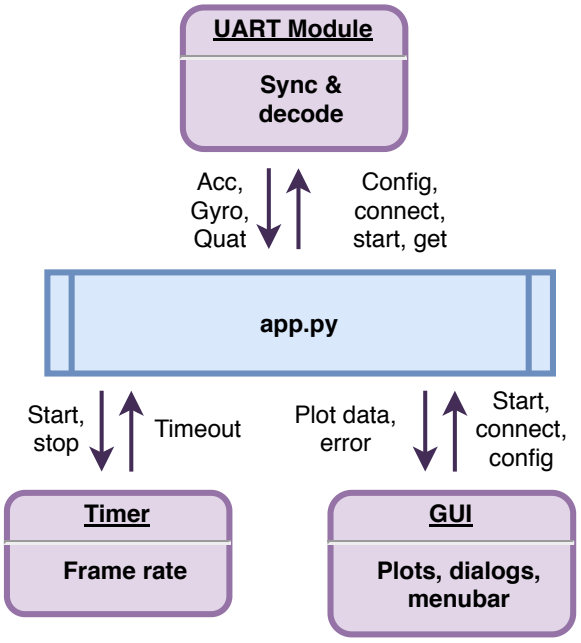


Figura 2: Diagrama de bloques del programa de la computadora

ledgement por parte del programa en la computadora, ya que los datos son enviados en tiempo real, y reenviar paquetes comprometería esta característica.

La sincronización se realizó estructurando los mensajes de la siguiente manera:

- 1 byte con la letra 'A' (por aceletrómetro) en ASCII
- 3 floats de 32 bits, indicando las componentes $x$, $y$ y $z$ del acelerómetro, en ese orden
- 1 byte con la letra 'G' (por giroscopio) en ASCII
- 3 floats de 32 bits, indicando las componentes $x$, $y$ y $z$ del giroscopio, en ese orden
- 1 byte con la letra 'Q' (por cuaternión en inglés, *quaternion*) en ASCII
- 4 floats de 32 bits, indicando las 4 componentes de la aproximación del cuaternión representando la posición de MPU

De esta manera, cada mensaje cuenta con 40 bytes de datos (3+3+4=10 floats, cada uno de cuatro bytes) y 3 bytes de sincronización. El programa en la computadora detecta la presencia de un mensaje válido a partir de estos tres caracteres en la posición adecuada. Esta tarea se realiza en un *thread* independiente, que constantemente busca nuevos mensajes y los guarda en una *queue*, hasta ser detenido por un evento de *stop*, o detectar un error en el puerto serie.

A continuación se muestra la totalidad del código correspondiente a este *thread*.

```
1     def uart_sync(self):
2       try:
3         # ignore all old msgs
4         self.ser.flush()
5
6         # wait for the next whole msg
7         buff = self.ser.read(MSG_SIZE)
8
9         # run until told to stop
10        while not self.end.is_set():
11
```

```
12            # check whether the msg fits the format
13            agq = buff[0::FLOAT_SIZE * 3 + 1][:3]
14            try:
15              agq = agq.decode(encoding='ascii')
16            except UnicodeDecodeError:
17              pass
18
19            # if the msg fits the format, queue it
20            if agq == 'AGQ':
21              self.q.put(buff)
22              buff = self.ser.read(MSG_SIZE)
23
24            # else drop first byte and read one more
25            else:
26              buff = buff[1:] + self.ser.read(1)
27
28        except (serial.SerialException, serial.SerialTimeoutException) as e:
29          # if there was a problem with the serial port, report it
30          self.q.put(e)
31        else:
32          # if no exceptions were raised, discard all previous msgs
33          self.q = Queue(maxsize=QUEUE_SIZE)
34        finally:
35          # mark event as read
36          self.end.clear()
37
```