```
const int serialBufferSize = 32;      // mida del buffer per
a entrades
char  serialBuffer[serialBufferSize]; // buffer per entrades
const int buttonPin = 2;              // numero del pin del
polsador
const int ledPin =  13;               // numero del pin del
led
const int serialMaxArgs = 4;          // max args del
missatge CSV
char* serialArgs[serialMaxArgs];      // args pointers
int outputTiming = 1000;              // temps enviament
paquet (ms) determina temps de sortida
int input1;                           // primer valor rebut
int input2;                           // segon valor rebut
int buttonState = 0;                  // variable estat del
polsador
int ledState = 0;                     // variable estat del
led

// Si definim input1 o input2 com a char ens tractara les
variables en codi ASCII
// Si ho definim com a int, treballem amb codi decimal

void setup(){

  Serial.begin(115200);
  pinMode(ledPin, OUTPUT);    // determinem el pin del led
com a sortida
  pinMode(buttonPin, INPUT);  // determinem el pin del
polsador com entrada

  // neteja del buffer de lectura (rx)
  while (Serial.available() > 0) { //mentre port obert llegir
i guardar en c
    byte c = Serial.read();
  }
 }


void loop(){
```

```
  static unsigned long  setupTime = millis();
  static unsigned long  loopTime = 0;
  static unsigned long  inputCount = 0;
  static unsigned long  outputCount = 0;
  static unsigned long  Time0 = 0;

  loopTime = millis() - setupTime; //Temps que dura el loop

  // Tasca de sortida
  // instruccions executades cada outputTiming ms (1seg)
  // el looptime sempre serà major a 0

  if (loopTime >= (outputCount * (unsigned
long)outputTiming)) {

   Serial.print ("XLS,get,Enviadades,B6"); //sol·licita dades
a l'excel
   Serial.print("\n");                    //acaba la
sol·licitud
  Serial.print ("XLS,get,Enviadades,B7"); //sol·licita dades
a l'excel
   Serial.print("\n");                    //acaba la
sol·licitud


  // Tasca d'entrada
  // Espera el missatge de resposta durant 50ms

  int len;

  while ((len = readLine((byte*)serialBuffer, 50)) > 0) {

    // mapejar els arguments CSV (valors separats per comes)
    // filtrar el missatge enviat per l'Excel per treure'n el
valor desitjat

   if (mapArgs(serialBuffer, len, serialArgs, serialMaxArgs)
> 0) {

      if (strcmp(serialArgs[0],"\0") == 0) {
       }
```

```
     else if (strcmp(serialArgs[0],"VAL") == 0)  {
        float x;
       x = atof(serialArgs[3]);  // x conté el valor rebut i
depen de la seqüència de demanda (FIFO)

       // 1a resposta basada en la referència del missatge
       if ((strcmp(serialArgs[1],"Enviadades") == 0) &&
(strcmp(serialArgs[2],"B6") == 0))  {
         input1 = x;                  // input1 conté el
valor de la fulla Enviadades cel·la B6
         Time0 = millis();            // obté el temps de
rebuda de la dada, var estatica
        }
        // 2a resposta
       if ((strcmp(serialArgs[1],"Enviadades") == 0) &&
(strcmp(serialArgs[2],"B7") == 0))  {
         input2 = x;                  // input2 conté el
valor de la fulla Enviadades cel·la B7

        }
        else  {
         // Resta d'entrades no sol·licitades
        }

      }
    }
    inputCount++;                     // suma +1 al contador
d'entrades
   clearBuffer((byte*)serialBuffer,serialBufferSize);

  }
        Serial.print("XLS,write,Enviadades,C6,");
         Serial.print(input1*1000);   // escriu en l'Excel
el valor input1 en ms
         Serial.print("\n");
        Serial.print("XLS,write,Enviadades,C7,");
         Serial.print(input2*1000);  // escriu en l'Excel el
valor input2 en ms
         Serial.print("\n");
        Serial.print("XLS,write,Enviadades,G4,");
         Serial.print(0);    // escriu en l'Excel un 0 per a
```

```
desactivar la connexió
         Serial.print("\n");

  }
   // Activació de la màquina
   // S'engega en el moment que el temps des de la dada és
obtinguda supera el temps
   // determinat en la dada1, s'apaga quan aquest temps
supera la dada2

     if(((millis()-Time0) > input1*1000 )&& ((millis()-
Time0)<input2*1000)){
       digitalWrite(ledPin,HIGH);
         }
      else{
        digitalWrite(ledPin,LOW);
        }

 // Llegir el estat del pulsador i del led:

 buttonState = digitalRead(buttonPin);
 ledState = digitalRead(ledPin);

 // comprovar si el pulsador està apretat o el led encés.
 // Definir com actuar en cada cas:

 if ((ledState == LOW)&& (buttonState == HIGH)) {
   // turn LED on:
  digitalWrite(ledPin, HIGH);
      }
 else { if ((ledState == LOW)&& (buttonState == HIGH)) {
   // turn LED ON:
  digitalWrite(ledPin, HIGH);
      }
  else {if ((ledState == HIGH)&& (buttonState == LOW)) {
   // turn LED off:
  digitalWrite(ledPin, LOW);
       }
 else  {
   // turn LED off:
  digitalWrite(ledPin, LOW);
```

```
        }}}


} // end loop

// rutina readLine
int readLine(byte* startAddr, int timeout) {

  int wait = 0;
  byte *p = startAddr;
  int receivedCount = 0;

  while (wait < timeout) {

    // compilar el buffer
    while (Serial.available() > 0) {

      *p = Serial.read(); // llegeix del port i emmagatzema

      // acaba la lectura si una linia acaba amb '\n'
      if (*p == '\n') {
        *p = '\0';
       return receivedCount;
      }
      receivedCount++;
      p++;
    }
    // esperar 1ms
    delay(1);
    wait++;
  }

  *p = '\0';
  return -1;
}


// mpateig dels arguments CSV (valors separats per comes)
int mapArgs(char* startAddr, int nBytes, char** argPointer,
int maxArgs) {
```

```
int  i;
char *p;

  *argPointer = startAddr;
  i = 1;

  for (p = startAddr; p < (startAddr + nBytes); p++) {
    if (*p == ',') {            // la coma separa els
diferents arguments

      *p = '\0';                // la coma es substitueix per
\0 aixi el buffer
                                // conté arguments llegibles
com strings
      if (i < maxArgs) {
        i++,
        argPointer++;
        *argPointer = p + 1;    // guardar l'adreça de
l'argument
      }
      else {

        return -1;
      }
    }
  }
  return i;
}


// neteja del buffer
  void clearBuffer(byte* startAddr, int nBytes) {

  byte   *p;

  for (p = startAddr; p < (startAddr + nBytes); p++) {
    *p = '\0';
  }
}
```