Arduino API

Contents

[Arduino Side – GenericRead.ino 3](#_Toc512962075)

[Functions 3](#_Toc512962076)

[Handshake Procedure 3](#_Toc512962077)

[Raspberry Pi Side – ArduinoComm.c 4](#_Toc512962078)

[SerialThreadArgs 4](#_Toc512962079)

[Defines 4](#_Toc512962080)

[Functions 5](#_Toc512962081)

[cURL 6](#_Toc512962082)

# Arduino Side – GenericRead.ino

The code for GenericRead.ino is designed to be used on several different Arduino devices, all connected to one Raspberry Pi. At a high level, the basic functionality of this file is to perform a handshake connection with the Raspberry Pi to inform the Pi of its location, then begin data collection by simply writing sensor values to the serial port.

## Functions

|  |  |  |  |
| --- | --- | --- | --- |
| Return Type | Name | Arguments | Description |
| void | setup | NULL | Initialize serial connection and perform a handshake connection |
| void | loop | NULL | Read sensor values and write them to the serial port |

## Handshake Procedure

1. Initialize serial communication and wait for connection
2. Wait for a single byte to be written from the Raspberry Pi
3. Flush the serial port
4. Write the location byte to the serial port

# Raspberry Pi Side – ArduinoComm.c

The ArduinoComm.c file is designed to connect to several Arduinos and read from their initial ports, remembering their locations as byte-codes, then start several threads to read from these Arduinos synchronously.

## SerialThreadArgs

typedef struct SerialThreadArgs {

char \*port;

FILE \*ser;

int loc;

int type;

pthread\_mutex\_t \*m;

} SerialThreadArgs;

SerialThreadArgs is a struct that allows each thread to have enough information to push to the database on its own. It members are as follows:

1. char \*port – a string providing the port name in the format “/dev/ttyXXXN” where N is a number
2. FILE \*ser – a pointer to the serial port as a file through which all communication is handled
3. int loc – an integer indicating the location of the Arduino for the thread, where each location should be unique
4. int type – an integer indication SINGLE\_DIR or DOUBLE\_DIR (see below)
5. pthread\_mutex\_t \*m – a mutex object protecting access to the cURL process across multiple threads, allowing each POST to be unique when sent to the database

## Defines

|  |  |  |
| --- | --- | --- |
| Name | Value | Description |
| NUM\_LOCATIONS | User-set (default 1) | Specifies the number of locations the Raspberry Pi should look for connections from |
| MAX\_LOCATIONS | 4 | The number of serial ports the Raspberry Pi can handle; this number can be extended via hardware hacking, but is most simply set to 4 |
| SINGLE\_DIR | 0 | Specifies a traffic entry that is uni-directional |
| DOUBLE\_DIR | 1 | Specifies a traffic entry that is bi-directional |
| START | 0x01 | Byte-code to send to the Arduino to initialize communication |
| TRIG\_DIST | User-set (default 400) | The distance in inches that the Raspberry Pi should use to determine whether or not it should send to POST |
| RATE | User-set (default 9600) | Baud rate of the serial communication with Arduinos; should be set to the same across all Arduinos and the Raspberry Pi |
| MAX\_BUF\_SIZE | 20 | Specifies a maximum buffer of 20 bytes |
| QUERY\_SIZE | 201 | Specifies the maximum buffer size of a query to send to POST |

## Functions

|  |  |  |  |
| --- | --- | --- | --- |
| Return Type | Name | Arguments | Description |
| int | main | NULL | Performs handshake procedure with Arduinos to initialize communication, then starts individual threads to collect data from Arduinos |
| void\* | serialThread | void \*args – a pointer to the argument struct | Uses information from args to decide which type of read each thread should be |
| void | ReadDoubleDir | SerialThreadArgs \*sta – a pointer to the argument struct | Reads traffic entry and exit when the entry and exit occur in the same lane |
| void | ReadSingleDir | SerialThreadArgs \*sta – a pointer to the argument struct | Reads traffic entry and exit when each lane only handles one direction |

## Handshake Procedure

1. Wait for serial communication to be initialized by Arduino
2. Write START
3. Read back the byte indicating location
4. Flush the serial port

## cURL

ArduinoComm.c uses cURL, a command line tool to send and receive data to/from a website. In this instance, cURL is used to send data to POST in a unique website that, upon startup, checks POST for a string. If that string exists, the website simply pushes it to the database; otherwise, the website quits and no change occurs.

cURL documentation and download can be found at <https://curl.haxx.se/>