Efficient communication between Arduino* and Linux* native processes

By Matthias H. (Intel) (https://software.intel.com/en-us/user/183576), Updated September 22, 2014

Working with Arduino* sketches on Intel® Galileo, or Intel® Edison you might run into situations where you would want to add some functionality coming from the underlying Yocto* Linux OS running underneath. Which brings us directly to the subject of this blog: How to efficiently communicate between those two worlds.Let's start to define some criteria we will try to meet in the following

Criteria

1. no communication on disk (SD card, eMMC) in order to reduce disk wearing and to improve performance
2. communication triggered by events, i.e. in particular we don't want to periodically check a status but to get informed by an event and idle otherwise

Inter Process Communication (IPC) on Linux

An Arduino* sketch running on Intel® Galileo, or Intel® Edison is in fact just a Linux process running in parallel to other Linux processes. As there is a full grown Linux running on the boards we can also use

standard means for inter process communication (IPC) between the Arduino* process and native process(es). There are various IPC methods on Linux. One out of those "memory mapped IPC". In essence it means that IPC processes share the same memory. Which means any modification from any process sharing that memory region will be at once visible to all the other processes. It also matches our first criteria to not write communication data on disk but operate only on memory.

Mutexes and condition variables

Having a shared memory leads to several questions like:

1. how to make sure only one process is operating on the shared data at a given time? (synchronization)
2. how to notify the other process(es) once data has been modified? (notification)

In the following we will look into those 2 questions one after the other. We will make use of "mutexes" and "condition variables" which are both contained in the POSIX threads (Pthreads) library available on Linux.

Synchronization - Mutex

A *mut*ual *ex*clusion (mutex) is a standard concept provided by probably any modern multitasking OS. In this blog we won't describe concepts and details here but only give high level information on POSIX threads (Pthreads) mutexes. For more information you should consult textbooks on operating systems (e.g. Tanenbaum, Woodhull: *Operating Systems 3rd ed*. Pearson 2006) or search online.

As the name says the Pthreads library focuses mainly on threads programming. However, it also offers powerful commands applicable to processes. Moreover the Arduino* IDE for Intel® Galileo, and Intel® Edison resp supports pthreads (i.e. links to the pthreads library) out of the box thus providing an easy integration. Hence, it seems a natural selection to use Pthreads for our purpose.

Typically a mutex makes sure a certain *critical region* of code is only accessed by one thread. As we are dealing with processes here we will use mutexes in a way that only one process can proceed a *pthread_mutex_lock* request in the code. Any other process will be set to sleep by the OS until *pthread_mutex_unlock* is called on the mutex after which the OS will wake all other processes requesting *pthread_mutex_lock.*

In pseudo code:

```
pthread_mutex_lock(&mutex);

// read / write shared memory data here

pthread_mutex_unlock(&mutex);
```

This has to be done in the same way for locking write as well as for locking read accesses as otherwise a read could access "half updated" data. In the next section we come to a further Pthreads concept which provides notification on data changes

Notification - Condition variable

Similar to the mutex concept where a process trying to access an already locked mutex Pthreads provides the concept of *condition variables*. The condition variable allows a thread, or in our case a process to request to be put to sleep until waked up through the variable. This is done by the function *pthread_cond_wait*.

Mutex and condition variable combined yield following pseudo code:

```
pthread_mutex_lock(&mutex);
pthread_cond_wait(&cond_variable, &mutex);

// read shared memory data here

pthread_mutex_unlock(&mutex);
```

The other process need to unlock the mutex and signal the change by calling *pthread_cond_signal*. This will wake up the sleeping process.

For further details pls check textbooks, or online tutorials resp on Pthreads. In the next section we will come to a sample implementation.

Implementation

some explanations:

- For the mutex as well as for the condition variable we need to explicitly set attributes in order to allow the inter process use
- We chose to utilize shared memory IPC via *memory mapped files* as the Arduino* IDE doesn't come with all libraries required for direct shared memory IPC. Putting the communication file in a filesystem mapped to main memory essentially provides identical functionality. The Yocto* Linux coming with Edison as well as the SD card Yocto* image on https://software.intel.com/iot have thetemp folder */tmp* mounted to *tmpfs* which lies in memory. I.e. any file within this folder would do. We chose the file "/tmp/arduino". It's meant to be used only for IPC.
- We assume that the Arduino process is the process to initilize the mutex and the condition variable as the Arduino process would start on system boot.
- we show only the situation where the Arduino* process waits for data from a Linux native process to operate on that data. For other purposes the code would have to be modified accordingly.
- to show the concept we put as data to share two booleans in the memory mapped structure *mmapData* which define whether the built-in and an external LED on IO 8 is to be switched on or off:

```
1 bool led8_on;   // led on IO8
2 bool led13_on;  // built-in led
```

  Obviously any other data could have been put there. The two other variables in the *mmapData* struct are the mutex and the condition variable

Note:

The sample code below is provided under the MIT license (http://opensource.org/licenses/MIT).

There are three files below:

- mmap.ino: sketch to be put onto Arduino* IDE
- mmap.cpp: native process sending data
- mmap.h: header file - same file to be used on Arduino* IDE as well as Linux native

I.E. on the Arduino* side you should have a folder which contains "mmap.ino" and "mmap.h" in your Arduino* sketches directory. On the Linux native side you should have a folder containing "mmap.cpp", and "mmap.h".

In order to run the sketch just open the "mmap" sketch in the Arduino* IDE and upload it to the right board (Intel® Galileo Gen 1, Intel® Galileo Gen 2, or Intel® Edison respectively). On the native side the Intel IoT devkit on https://software.intel.com/iot comes with a cross compiler, or alternatively the Yocto* Linux coming with Intel® Edison as well as the SD card Yocto* image fon https://software.intel.com/iot or Intel® Galiileo come with a C++ compiler pre-installed. Using the pre-installed compiler you would need to run

```
1 g++  mmap.cpp   -lpthread -o mmap
```

in the folder you put mmap.cpp and mmap.h. It will generate a binary you can execute as follows

```
1 ./mmap {0,1}{0,1}
```

where "{0,1}" stands for either 0 or 1. I.E. "./mmap 00" would switch both LEDs off, whereas "./mmap 11" would switch both LEDs on. Some additional output to the serial monitor [CTRL+SHIFT+M] on the Arduino* IDE, and to the console you start the Linux native process show in addition the data which has been set.

mmap.ino

```
01 /*
02  * Author: Matthias Hahn <matthias.hahn@intel.com>
03  * Copyright (C) 2014 Intel Corporation
```

```
 * This file is part of mmap IPC sample provided under the MIT license
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:

 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.

 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */

#include "mmap.h"

using namespace std;

/* assume /tmp mounted on /tmpfs -> all operation in memory */
/* we can use just any file in tmpfs. assert(file size not modified && file permissions left readable) */
struct mmapData* p_mmapData; // here our mmapped data will be accessed

int led8 = 8;
int led13 = 13;


void exitError(const char* errMsg) {
  /* print to the serial Arduino is attached to, i.e. /dev/ttyGS0 */
  string s_cmd("echo 'error: ");
  s_cmd = s_cmd + errMsg + " - exiting' > /dev/ttyGS0";
  system(s_cmd.c_str());
  exit(EXIT_FAILURE);
```

```
43 }
44
45 void setup() {
46   int fd_mmapFile; // file descriptor for memory mapped file
47   /* open file and mmap mmapData*/
48   fd_mmapFile = open(mmapFilePath, O_CREAT | O_RDWR, S_IRUSR | S_IWUSR);
49   if (fd_mmapFile == -1) exitError("couldn't open mmap file");
50   /* make the file the right size - exit if this fails*/
51   if (ftruncate(fd_mmapFile, sizeof(struct mmapData)) == -1)
exitError("couldn' modify mmap file");
52   /* memory map the file to the data */
53   /* assert(filesize not modified during execution) */
54   p_mmapData = static_cast<struct mmapData*>(mmap(NULL, sizeof(struct
mmapData), PROT_READ | PROT_WRITE, MAP_SHARED, fd_mmapFile, 0));
55   if (p_mmapData == MAP_FAILED) exitError("couldn't mmap");
56   /* initialize mutex */
57   pthread_mutexattr_t mutexattr;
58   if (pthread_mutexattr_init(&mutexattr) == -1)
exitError("pthread_mutexattr_init");
59   if (pthread_mutexattr_setrobust(&mutexattr, PTHREAD_MUTEX_ROBUST) == -1)
exitError("pthread_mutexattr_setrobust");
60   if (pthread_mutexattr_setpshared(&mutexattr, PTHREAD_PROCESS_SHARED) == -1)
exitError("pthread_mutexattr_setpshared");
61   if (pthread_mutex_init(&(p_mmapData->mutex), &mutexattr) == -1)
exitError("pthread_mutex_init");
62
63   /* initialize condition variable */
64   pthread_condattr_t condattr;
65   if (pthread_condattr_init(&condattr) == -1)
exitError("pthread_condattr_init");
66   if (pthread_condattr_setpshared(&condattr, PTHREAD_PROCESS_SHARED) == -1)
exitError("pthread_condattr_setpshared");
67   if (pthread_cond_init(&(p_mmapData->cond), &condattr) == -1)
exitError("pthread_mutex_init");
68
69   /* for this test we just use 2 LEDs */
70   pinMode(led8, OUTPUT);
71   pinMode(led13, OUTPUT);
72 }
73
74 void loop() {
75   /* block until we are signalled from native code */
76   if (pthread_mutex_lock(&(p_mmapData->mutex)) != 0)
exitError("pthread_mutex_lock");
```

```
77    if (pthread_cond_wait(&(p_mmapData->cond), &(p_mmapData->mutex)) != 0)
exitError("pthread_cond_wait");
78
79    if (p_mmapData->led8_on) {
80      system("echo 8:1 > /dev/ttyGS0");
81      digitalWrite(led8, HIGH);
82    }
83    else {
84      system("echo 8:0 > /dev/ttyGS0");
85      digitalWrite(led8, LOW);
86    }
87    if (p_mmapData->led13_on) {
88      system("echo 13:1 > /dev/ttyGS0");
89      digitalWrite(led13, HIGH);
90    }
91    else {
92      system("echo 13:0 > /dev/ttyGS0");
93      digitalWrite(led13, LOW);
94    }
95    if (pthread_mutex_unlock(&(p_mmapData->mutex)) != 0)
exitError("pthread_mutex_unlock");
96  }
```

mmap.cpp

```
01  /*
02   * Author: Matthias Hahn <matthias.hahn@intel.com>
03   * Copyright (C) 2014 Intel Corporation
04   * This file is part of mmap IPC sample provided under the MIT license
05   *
06   * Permission is hereby granted, free of charge, to any person obtaining a
copy
07   * of this software and associated documentation files (the "Software"), to
deal
08   * in the Software without restriction, including without limitation the
rights
09   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
10   * copies of the Software, and to permit persons to whom the Software is
11   * furnished to do so, subject to the following conditions:
12   *
13   * The above copyright notice and this permission notice shall be included in
14   * all copies or substantial portions of the Software.
```

```
15
16   * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
17   * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
18   * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
19   * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
20   * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
     FROM,
21   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
22   * THE SOFTWARE.
23   */
24
25  /* mmap.cpp
26     Linux native program communicating via memory mapped data with Arduino
     sketch.
27     Compilation: g++  mmap.cpp   -lpthread -o mmap
28     Run: ./mmap <LED8><LED13> (e.g. ./mmap 01 -> LED 8 off, LED 13 on)
29     For "random" blink you may run following commands in the command line:
30     while [ 1 ]; do ./mmap $(($RANDOM % 2))$(($RANDOM % 2)); done
31  */
32
33  #include "mmap.h"
34
35  void exitError(const char* errMsg) {
36    perror(errMsg);
37    exit(EXIT_FAILURE);
38  }
39
40
41  using namespace std;
42
43
44  /**
45   * @brief: for this example uses a binary string "<led8><led13>"; e.g. "11":
     both leds on
46   * if no arg equals "00"
47   * For "random" blink you may run following commands in the command line:
48   * while [ 1 ]; do ./mmap $(($RANDOM % 2))$(($RANDOM % 2)); done
49   */
50  int main(int argc, char** argv) {
51    struct mmapData* p_mmapData; // here our mmapped data will be accessed
52    int fd_mmapFile; // file descriptor for memory mapped file
53
54    /* Create shared memory object and set its size */
55    fd_mmapFile = open(mmapFilePath, O_CREAT | O_RDWR, S_IRUSR | S_IWUSR);
```

```
56    if (fd_mmapFile == -1) exitError("fd error; check errno for details");
57
58    /* Map shared memory object read-writable */
59    p_mmapData = static_cast<struct mmapData*>(mmap(NULL, sizeof(struct
      mmapData), PROT_READ | PROT_WRITE, MAP_SHARED, fd_mmapFile, 0));
60    if (p_mmapData == MAP_FAILED) exitError("mmap error");
61    /* the Arduino sketch might still be reading - by locking this program will
      be blocked until the mutex is unlocked from the reading sketch
62     * in order to prevent race conditions */
63    if (pthread_mutex_lock(&(p_mmapData->mutex)) != 0)
      exitError("pthread_mutex_lock");
64    if (argc == 1) {
65      cout << "8:0" << endl;
66      cout << "13:0" << endl;
67      p_mmapData->led8_on = false;
68      p_mmapData->led13_on = false;
69    }
70    else if (argc > 1) {
71      // assert(correct string given)
72      int binNr = atol(argv[1]);
73      if (binNr >= 10) {
74        cout << "8:1" << endl;
75        p_mmapData->led8_on = true;
76      }
77      else {
78        cout << "8:0" << endl;
79        p_mmapData->led8_on = false;
80      }
81      binNr %= 10;
82      if (binNr == 1) {
83        cout << "13:1" << endl;
84        p_mmapData->led13_on = true;
85      }
86      else {
87        cout << "13:0" << endl;
88        p_mmapData->led13_on = false;
89      }
90    }
91    // signal to waiting thread
92    if (pthread_mutex_unlock(&(p_mmapData->mutex)) != 0)
      exitError("pthread_mutex_unlock");
93    if (pthread_cond_signal(&(p_mmapData->cond)) != 0)
      exitError("pthread_cond_signal");
94  }
```

mmap.h

```
01 /*
02  * Author: Matthias Hahn <matthias.hahn@intel.com>
03  * Copyright (C) 2014 Intel Corporation
04  * This file is part of mmap IPC sample provided under the MIT license
05  *
06  * Permission is hereby granted, free of charge, to any person obtaining a
    copy
07  * of this software and associated documentation files (the "Software"), to
    deal
08  * in the Software without restriction, including without limitation the
    rights
09  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
10  * copies of the Software, and to permit persons to whom the Software is
11  * furnished to do so, subject to the following conditions:
12  *
13  * The above copyright notice and this permission notice shall be included in
14  * all copies or substantial portions of the Software.
15  *
16  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
17  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
18  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
19  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
20  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM,
21  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
22  * THE SOFTWARE.
23  */
24
25 #ifndef MMAP_HPP
26 #define MMAP_HPP
27
28 #include <fcntl.h>
29 #include <unistd.h>
30 #include <sys/mman.h>
31 #include <iostream>
32 #include <string.h>
33 #include <stdlib.h>
34 #include <cstdio>
35 #include <pthread.h>
36
```

```
37 /* assert(/tmp mounted to tmpfs, i.e. resides in RAM) */
38 /* just use any file in /tmp */
39 static const char* mmapFilePath = "/tmp/arduino";
40
41
42 struct mmapData {
43   bool led8_on;   // led on IO8
44   bool led13_on;  // built-in led
45   pthread_mutex_t mutex;
46   pthread_cond_t cond;
47 };
48
49 #endif
```


23 comments

Top

Asher F. (/en-us/user/1412367) said on Jun 4,2016

(/en-us/user/1412367)

Thanks, very helpful.

Matthias H. (Intel) (/en-us/user/183576) said on Mar 18,2016

(/en-us/user/183576)

@Chinedu O. (https://software.intel.com/en-us/user/1253329)

Just downloaded latest Arduino IDE v. 1.6.8 and the corresponding Intel(R) Edison board packages (haven't checked for Intel(R) Galileo though). Under

C:\Users\<username>\AppData\Local\Arduino15\packages\Intel\tools\core2-32-poky-linux\1.6.2+1.0\core2-32-poky-linux\usr\include

I can see all the header files. So I guess something in your setup is messed up?

Chinedu O. (/en-us/user/1253329) said on Mar 9,2016

(/en-us/user/1253329)

I like the article but I am having one problem. When I include the mmap.h to my arduino sketch. I get the error about all the header files included in them. I get that non of them is included in the directory. Am I doing something wrong? Please help I am new to arduino!

Matthias H. (Intel) (/en-us/user/183576) said on Aug 24,2015

(/en-us/user/183576)

@Bao N

It looks like you try to share a pointer to a char array? That's not gonna work that way. How should one process be able to access the address space of the other process (unless they are in shared memory space)?

In your case I guess the simplest solution would be to replace

```
char *data = NULL;
```

by

```
char data[16];
```

Matthias H. (Intel) (/en-us/user/183576) said on Aug 24,2015

(/en-us/user/183576)

@Sandesh K

> i want to run a python script when arduino part triggers an event , and i want to get the result of that script execution in arduino part . How to make it possible ? Afters hours of browsing i landed no where. Please help me over it

As far as I understand this request it's a "ping-pong" like design:

1. Arduino triggers Python script running
2. Arduino waiting for response
3. back to 1

For 1 you need to implement the other way round: "native" side waiting for Arduino sending. For 2 you might use the approach described in the blog.

Bao N. (/en-us/user/1183759) said on Aug 22,2015

(/en-us/user/1183759)

Hi matthias-hahn,

I'm verry happy to read your article. Your sample code run sussesfully in my Intel Galileo Gen2 board.

But a segfault error happen when I tried to use dynamic char array in p_mmapData which is initialized by malloc().

This is my modified declaration for p_mmapData.

```
struct mmapData
{
    bool newData = false;
    char *data = NULL;
    pthread_mutex_t mutex;
    pthread_cond_t cond;
};
```

This is my modified code in mmap.cpp

```
if (argc != 1)
{
        p_mmapData->newData = true;
        p_mmapData->data = (char*)malloc(17);   // I want to display data by a 16x2 LCD
        strcpy(p_mmapData->data, argv[1]);
}
```

This is my modified code in mmap.ino

```
if (p_mmapData->newData)
{
        lcd.clear();  // clear LCD
        lcd.print(p_mmapData->data); // display on LCD
        free(p_mmapData->data);
        p_mmapData->newData = false;
}
```

Thanks a lots for help.

Sandesh K. (/en-us/user/1173282) said on Jul 12,2015

(/en-us/user/1173282)

i want to run a python script when arduino part triggers an event , and i want to get the result of that script execution in arduino part . How to make it possible ? Afters hours of browsing i landed no where. Please help me over it

Matthias H. (Intel) (/en-us/user/183576) said on Jan 26,2015

(/en-us/user/183576)

> "Do you have any suggestions for being able to pass messages of an arbitrary size? I'm planning to use a fixed-size buffer and a "bytes_remaining" field in the mmapData struct, but wondering if you have a suggestion for a better way."

You need to know the memory map structure. So you'd have to unmap&remap and sync the new size between processes... - don't think that would be very efficient. Rather you may split your data into blocks of the defined size (and a remainder for the last bytes) - surely you'd have to invent some control information (# of consecutive blocks ...) but I guess that would make more sense.

Dick F. (/en-us/user/1123767) said on Jan 26,2015

(/en-us/user/1123767)

Hi Matthias,

Do you have any suggestions for being able to pass messages of an arbitrary size? I'm planning to use a fixed-size buffer and a "bytes_remaining" field in the mmapData struct, but wondering if you have a suggestion for a better way.

Best,

Dick F

Matthias H. (Intel) (/en-us/user/183576) said on Nov 26,2014

(/en-us/user/183576)

Hi Parthiban,

@1: you don't need to use Arduino. Just you may want to. If yes, you may want to communicate with other processes. . IInmy last reply I already tried to explain that there is no emulator. There are just wrapper functions. At the end the Arduino IDE cross compiles to a Linux binary and sends it over to the target where it is run from clloader service.

@2: don't think so. It's a certain subset which is sufficient to run Arduino libs. E.G. the reason to go via a file in tmpfs rather than on /dev/shm is that some support for /dev/shm was missing in the IDE version I looked onto

@3: parts of it are definitely open source - just download the IDE and have a look for licenses, ... on the background see #1

Add a Comment

Sign in (/en-us/user/login?language=en-us&destination=node/531744&https=1)

Have a technical question? Visit our forums (/en-us/forum).
Have site or software product issues? Contact support (/en-us/contact).

- Connect
  - Collaborate on Dev Mesh
  - Instructables.com
- Related Resources
  - Intel® Quark™ Processors
  - Intel® IoT Gateway Technology

- Hackster.io

- Intel® Embedded Design Center
- Intel® RealSense™ Technology
- Makers and Intel

- **Verticals**

- Automated Driving
- Retail

Get the Newsletter

Follow us: