**Communication protocols**

**Terms:**

Brick: The physical “LEGO” block

Bridge: Software in charge of gathering the state of each **Brick**.

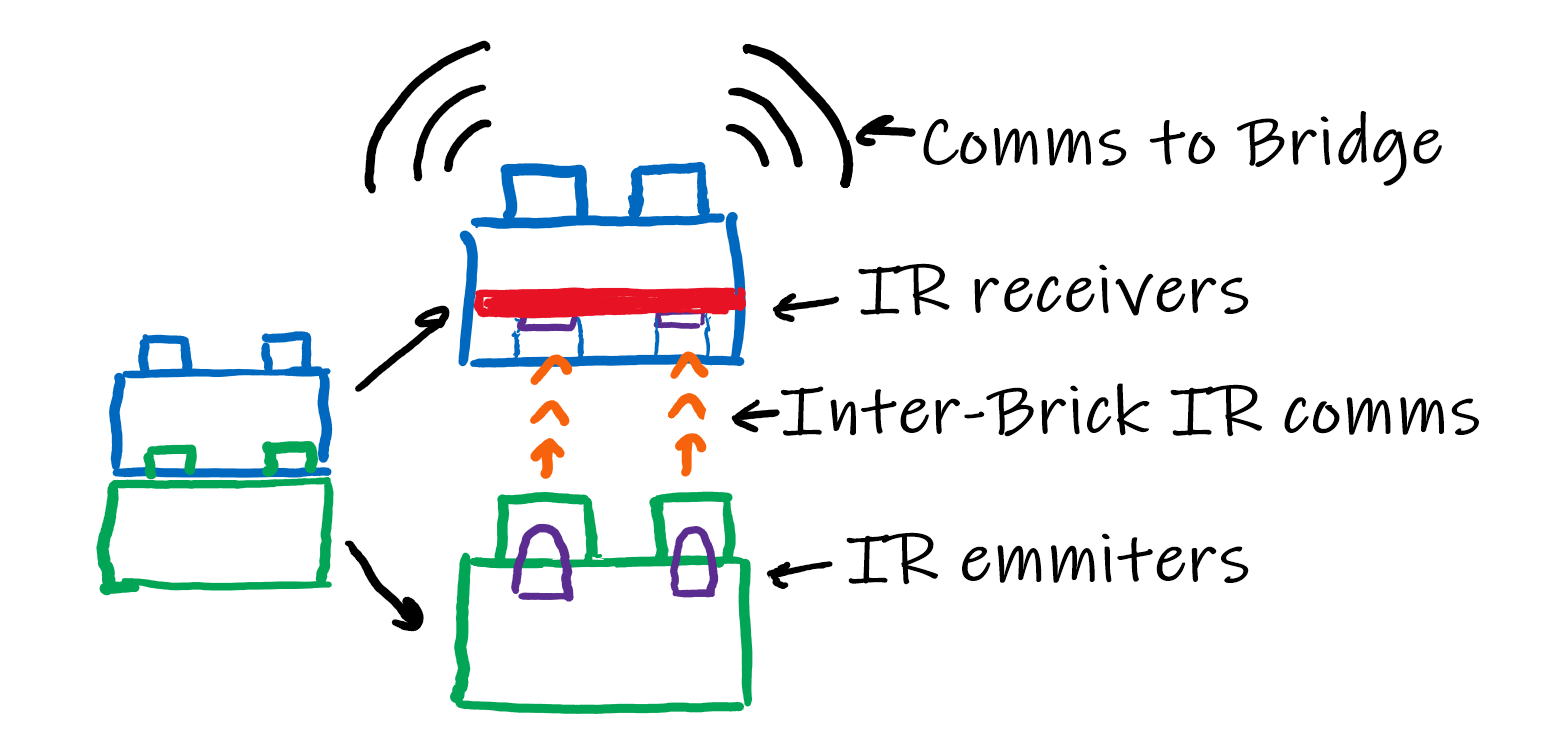
This part changes depending on the communication technology/OS used and provides a hardware agnostic abstraction layer to the **Builder**.

Builder: It reconstructs the building from the block states coming from the **Bridge.**

CAD: Takes the building data from the **Builder** and displays it accordingly.

**Protocol descriptions:**

1. Brick to Brick



The Brick to Brick communication is necessary so that each brick knows which other bricks it is connected to.

This communication is unidirectional to save on costs and to make the transmitters/receivers fit the holes. Therefore, a brick only knows about the bricks connected below it.

In order to reduce the necessary MCU resources (RAM), data from previous bricks is not transferred along the “chain”. Thanks to this, only the data about the blocks directly below is stored. This data will later be broadcasted through a wireless connection to the Bridge.

**Communication protocol**

During the active mode, each Brick will be constantly sending data packets through the IR transmitters on each LED.

The transmission interval between packets will be defined during experimentation to save power, but anything in between 200ms-500ms should work fine.

The frequency at which the data is transmitted will likely be quite low (100Hz-2kHz), which allows us to clock the MCUs slower to save power and to have a longer margin to read the incoming data as the communication is asynchronous. (The data clocks of the different bricks won’t necessarily be in sync, and therefore a much faster sampling rate than the transmitting clock rate is needed).

**Standard packet structure**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | B3 | B2 | B1 | B0 | - | - | - | - |
| Start of packet (Fixed) | | | | | | | | Packet type | | | | RESERVED | | | |

**DATA BYTES HERE**

**\/**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Data bytes length | | | | | | | | CRC | | | | | | | |

**Heartbeat packet**

Packet ID: 0, 0x0, B\_0000

Sent each interval to signal the block above that it is connected to another “smart” block.

Even though the *Identification packet* could also be used as a heartbeat, the lower size and higher frequency of this packet makes it a better choice.

**Identification packet**

Packet ID: 1, 0x1, B\_0001

This packet contains data about the block sending it.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Block ID Size (In Bytes) | | | | | | | | Block ID (Byte X) x size | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Sending Pin ID Size (In bytes) | | | | | | | | Sending Pin ID (Byte X) x size | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Block type (Byte 1) | | | | | | | | Block type (Byte 0) | | | | | | | |

1. Brick to Bridge

To avoid limiting the number of blocks that can be connected to the system, a connectionless system will be used.

Apart from that, to eliminate the need of using an additional piece of hardware to receive/send the wireless data on the Bridge side, Bluetooth low energy (BLE) is used. Most laptops and smartphones now include a bluetooth radio and therefore they can be used to connect to the Bricks.

This is also one of the motivations to separate the Bridge from the Builder software, allowing the end-user to choose if they want to use a laptop as the Bridge or their Smartphone.

**BLE Stack**

As we wanted to only use the connectionless features of BLE, the only available ones are:

* Advertisements: Packets sent at “regular” intervals that contains a source MAC address and up to 31 bits of data.
* Scan requests/responses: Sent after an advertisement. Used to increase the amount of data an advertisement has.

The biggest downside of having each Brick broadcast (advertise) its own data is that if all the Bricks are broadcasting at the same time, no data will go through successfully. Therefore, a way to reduce the amount of “air-time” for each brick is needed.

**Reducing the Brick “air-time”**

1. Bridge <-> Builder
2. Builder <-> CAD

After examining the possible options for a CAD tool, Bricklink Studio 2.0 seems like the best one. It has the best UI and other features that would be interesting for the project (Creating instructions for the model and a huge customizable part library).

Due to it being closed-source, editing its source code is not an option. Therefore, we need to inject our code (in a DLL) at runtime. This code should allow the Builder program to place/remove blocks and run any other functions.

Thanks to it being a Unity “game” and therefore running a .NET Mono assembly, it is easy to decompile the assembly to see which methods we want to interface to.

After importing both the Unity and the CAD tool assemblies into Visual Studio, we can create a Library (.dll) and write C# code as usual.

Finally, to load our DLL into the running software, we need to inject the DLL into the CAD process. This is automatically performed by our software once the CAD editor is detected running. (An open-source injector, SharpMonoInjector, is used to perform the injection itself)

Studio CAD stuff:

**Brick positioning:**

The x and z coords are the ones in the center of the brick

The y coordinate is the coord of the top of the brick. A more negative y means higher.

Each 1x1 stud position is 20x20 in size

Each 1 of brick height is 24 units

**Useful Brick IDs:**

3001 -> 4x2

3003 -> 2x2

702 -> 4x4 corner

Useful