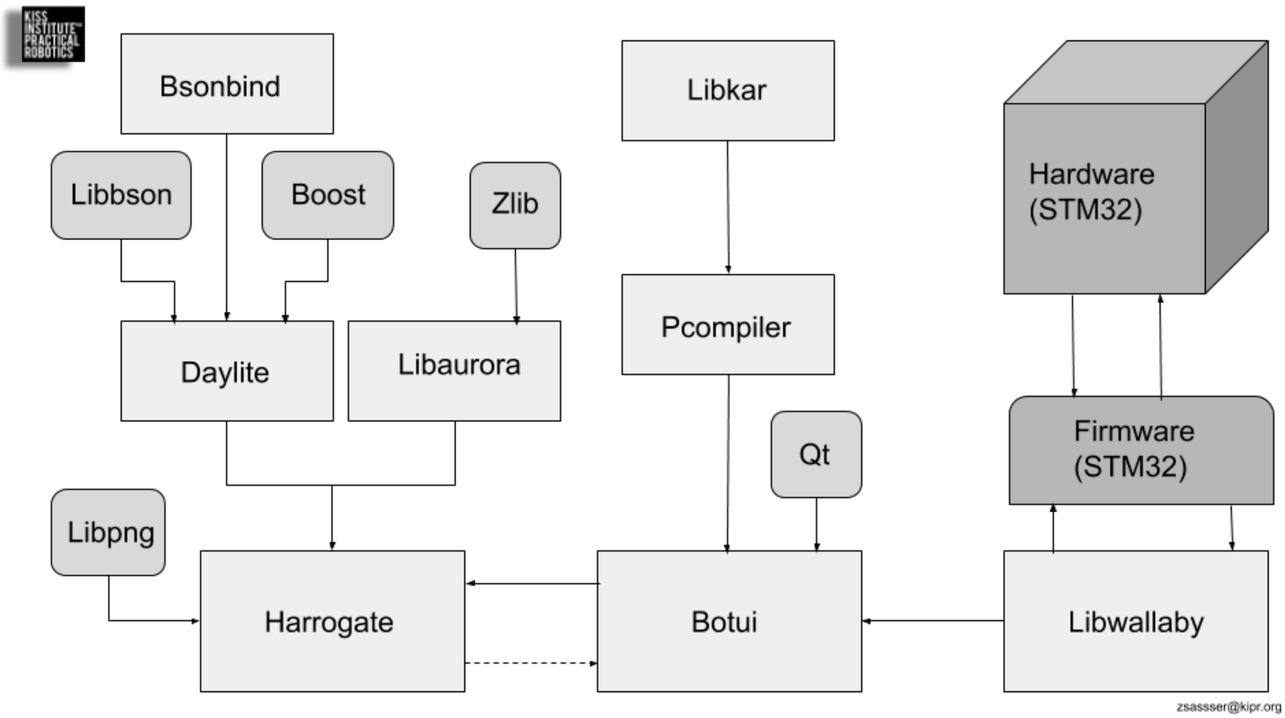
Wombat Developer Manual

Author(s): Zachary Sasser

Overview of KIPR-Suite



Program	Description
Libwallaby	Library for interfacing with the controller hardware.
Botui	UI for Interfacing with the controller
Harrogate	Node.js webserver for KISS-IDE
Pcompiler	Used to compile KISS-IDE projects
Libkar	Archiving tool used by Pcompiler
Daylite	Networking Backbone for Harrogate
Libaurora	Graphics Library that interfaces with Daylite
Bsonbind	Utility that generates C++ structs for descriptions of BSON

Note: These program descriptions are over-simplified, visit their respective github pages/documentation for more information

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Wombat Developer Manual

Operating System Installation:

Linux/MacOS/Unix:

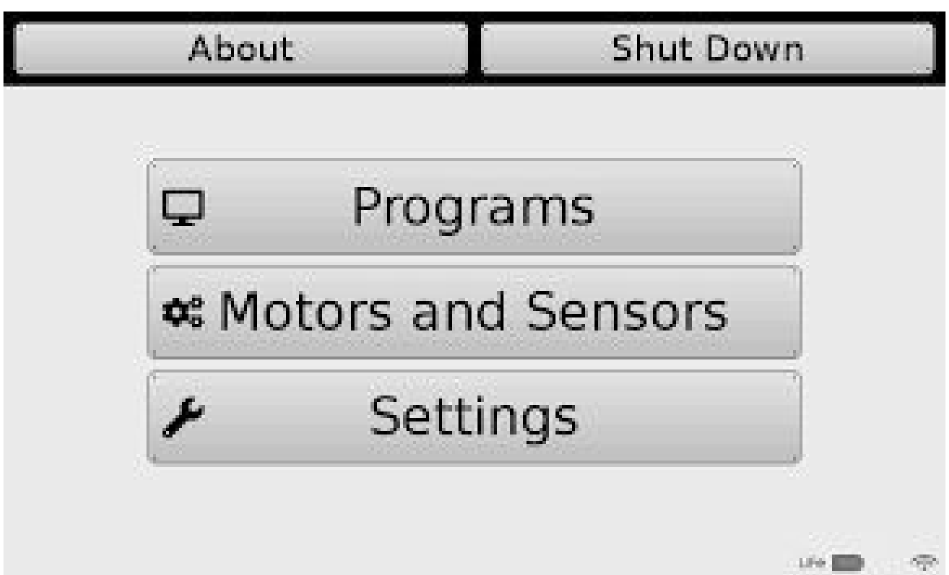
1. Download the latest KIPR-OS image: [Click Here (Dropbox)]
2. Most Linux Distros come with dd installed, but if not run the following:
    a. "sudo apt-get install gdebi"
3. Insert an SD card and find which pseudo-directory it is.
    a. "sudo fdisk -l"
    b. Alternate: "sudo df"
4. If your drive is /dev/sdb (/dev/sdb1 is a partition of sdb), then you would type
    a. "sudo dd if="Wombat.img" of=/dev/sdb bs=4M status=progress

Windows (or Linux/MacOS):

1. Download Balena Etcher and install it to your PC: https://www.balena.io/etcher/
2. Download the latest KIPR-OS image: [Click Here (Dropbox)]
3. Open Balena Etcher and follow the simple prompts, choose your Wombat image and your SD card

On Startup:

The first time you boot up your pi, it will run fsck which is similar to CHKDSK in MS-DOS. Then it will reboot and boot into this screen; this is Botui. To get to the desktop, go to "*Settings*□*Hide UI*" or hit "*WIN+D*".



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Wombat Developer Manual

Flashing the STM32:

When the Wombat is first assembled or the STM32 gets out of sync (DMA Desynchronized), you have to write the firmware binary to the STM32 processor (the device that handles the hardware). To flash the processor, navigate to /home/pi and then type “sudo ./wallaby_flash”, The output should look like this:

```
Wombat:~ $ sudo ./wallaby_flash
/sys/class/gpio/gpio17 - initializing gpio pins
echo 1 > /sys/class/gpio/gpio17/value
resetting co processor...
stm32flash -v -S 0x08000000 -w /home/pi/wallaby_v8.bin /dev/ttyAMA0
stm32flash 0.4

http://stm32flash.googlecode.com/

Using Parser : Raw BINARY
Interface serial_posix: 57600 8E1
Version      : 0x31
Option 1     : 0x00
Option 2     : 0x00
Device ID    : 0x0419 (STM32F427/37)
- RAM        : 192KiB  (8192b reserved by bootloader)
- Flash      : 1024KiB (sector size: 4x16384)
- Option RAM : 16b
- System RAM : 29KiB
Write to memory
Erasing memory
Wrote and verified address 0x080030f4 (100.00%) Done.

echo 0 > /sys/class/gpio/gpio17/value
resetting co processor...
```

If you are getting the “/sys/class/gpio/export: Permission denied” error:

- Try “sudo chmod +x wallaby_*” or “sudo chmod 777 wallaby_flash”
  - (Security doesn’t really matter for this)
- You are in the wrong terminal (use root not tty#)
- The GPIO cannot connect to the STM32 (usually because of a race condition)
- There is something blocking the output of the GPIO pins

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Wombat Developer Manual

Botui Project Structure:

Folder	Function
src ("source")	This is for all the C++ files that handles any specific page's backend. The files are organized by the title of the page. The title in the src file should match the accompanying UI and include files (or anything really)
include/botui	This file is just a collection of header files. Qt objects have their function definitions here.
ui	This is for storing the XML/HTML-like .ui files that Qt uses for it's display. You can edit these files using software called Qt Creator. It is also possible to do it by hand with a text editor if you are used to dealing with markup languages.
Rc ("Resource Collection")	RC is a file that is used for Qt icons, fonts, and other things like that. It allows us to use the Qt icon database for things like buttons, and technically also handles the font.
ts ("Translation Strings")	The ts files stores all the translations for text. Whenever we get a translator for a language, they will go and edit this file to replace it with the target language.
devices	These are header files that allow other parts of the project to access certain things on the controller. This is things from battery level to the serial number (stored on I2C registers). This is not how the program handles physical I/O like motors and digital ports. Libwallaby is responsible for handling the interaction with the STM32 directly.
dbus ("Desktop Bus")	Dbus is a software bus that allows you to communicate directly between processes. Botui (as of writing this) only uses it to access the network manager. This handles the access point that is broadcast by the Wombat.
debian	This is for building the project into a package

```
#Installing dependencies:
sudo apt-get install libzbar-dev libopencv-dev libjpeg-dev python-dev doxygen swig

# Create a build directory (inside botui) and change directory to it
mkdir build
cd build

# Initialize CMake
cmake ..

# Build
make

# Install
sudo make install
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Wombat Developer Manual

Libwallaby Project Structure:

Folder	Function
src (“source”)	This is for all the C++ files that handle the direct interaction with the STM32, translating the simplified functions in the library to the complexity of hardware.
include	The include folder is split into KIPR and Wallaby. KIPR is the modern version for the Wombat and potential future controllers. Wallaby is the elder used by the Wallaby.
bindings	The way that python is ran on the controller, is by “compiling” it down to C using SWIG. The bindings file contains SWIG related files that are used for this process.
Doxygen	Doxygen is used to generate the help page documentation. When you add comments to a function added to the library, it automatically generates the item on the help page. Here is a link to the version that KIPR hosts (old): **https://www.kipr.org/doc/index.html**
debian	This is for building the project into a package
cmake	Stores extra cmake files when needed.
tests	These are unit testing programs.

Note: Harrogate is not included as it is mostly self explanatory in respect to Nodejs.

```
#Installing dependencies:
sudo apt-get install libzbar-dev libopencv-dev libjpeg-dev python-dev doxygen swig

# Create a build directory (inside libwallaby) and change directory to it
mkdir build
cd build

# Initialize CMake
cmake ..

# Build
make

# Install
sudo make install
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Wombat Developer Manual

How to build a Project

```
cd <Project Folder>
mkdir build
cd build
cmake ..
make -j4
sudo make install <project name>
```

<Project Folder>: The folder downloaded from github, the main folder for the project.
make -j4: Adding -j4 to make allows the compiler to use 4 threads, you can adjust this number.

How to convert to a (.deb)

```
tar -zcvf <project>.tar.gz <project folder>
cp <project>.tar.gz <project folder>
cd <project>
dpkg -buildpackage -b -rfakeroot -us -uc
```

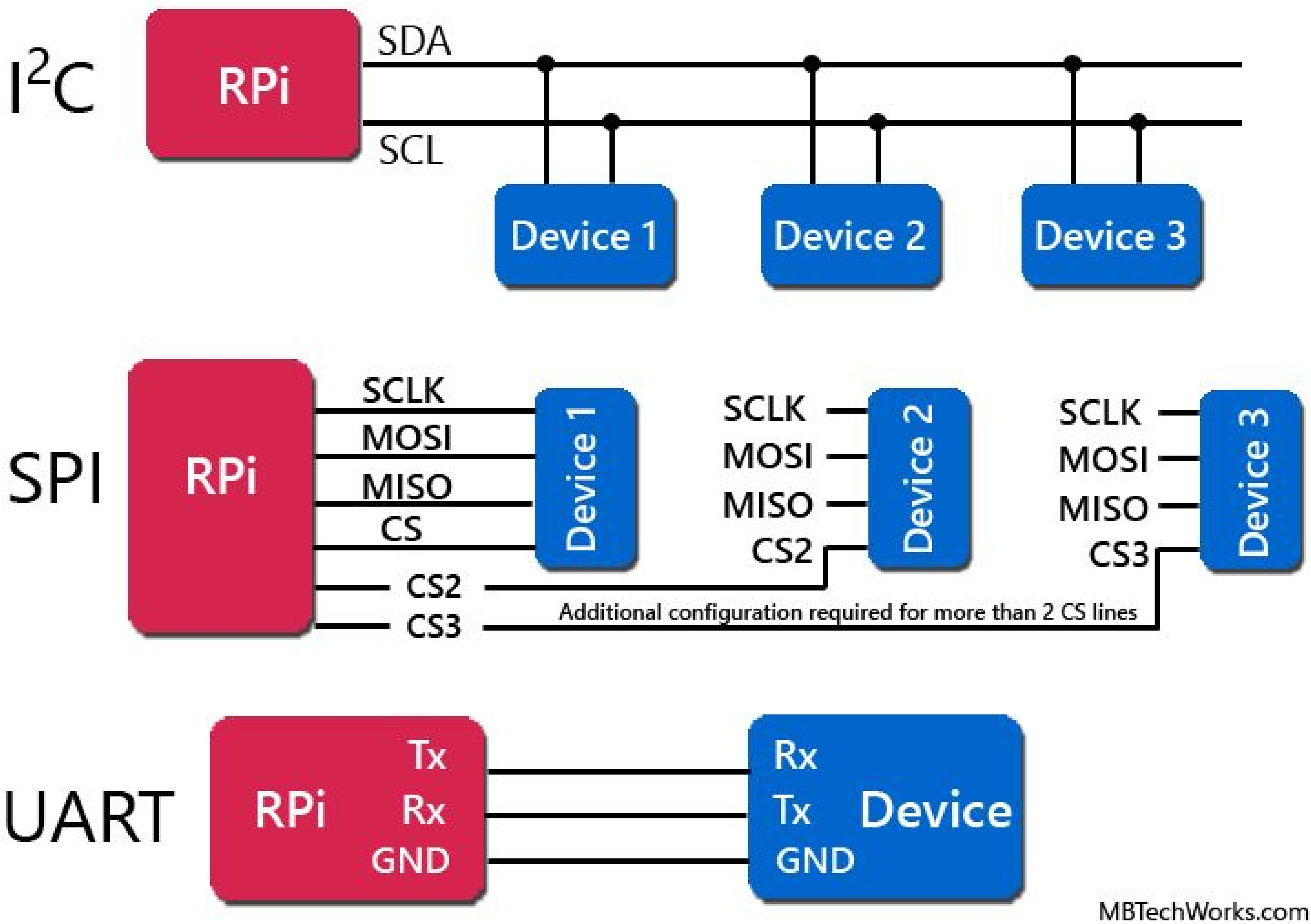
Creating a (.img) of your OS State

1. Take the SD card out of the Wombat (from the Pi, can be accessed at the bottom of the case)
2. Plug the SD card into a Linux PC and navigate to terminal
3. Find the correct drive with “sudo fdisk -l” (l for “list”)
4. Use the command below to save to output to a (.img) or a (.bin)
5. *(optional)* Install [PiShrink.sh] and run “sudo pishrink.sh Wombat.img”

```
sudo dd if=/dev/sdb of=”Wombat.img” bs=4M status=progress
```

if □ “input file”
of □ “output file”
bs □ “block size”
status=progress □ Show a progress bar/numbers

Communication Protocols



The Wombat uses multiple communication protocols to access the STM32 as well as other devices. For example, the I2C lines are used to access the Touchscreen controller.

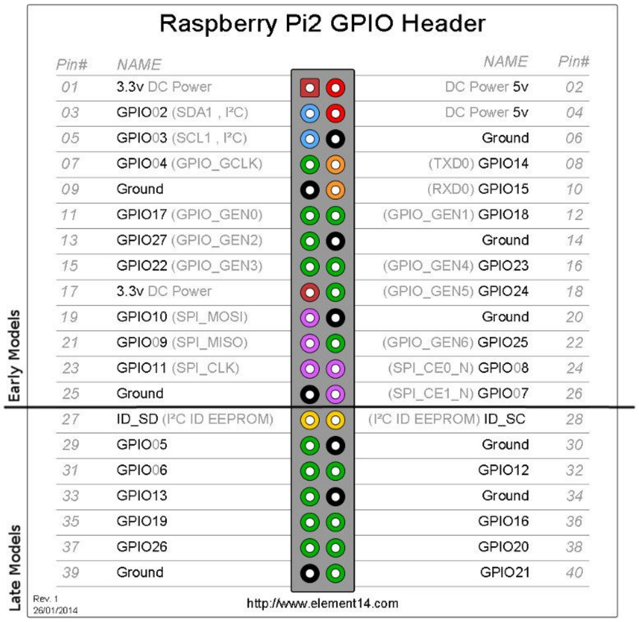
This is a good source for more information on these technologies:
https://www.seeedstudio.com/blog/2019/09/25/uart-vs-i2c-vs-spi-communication-protocols-and-uses/

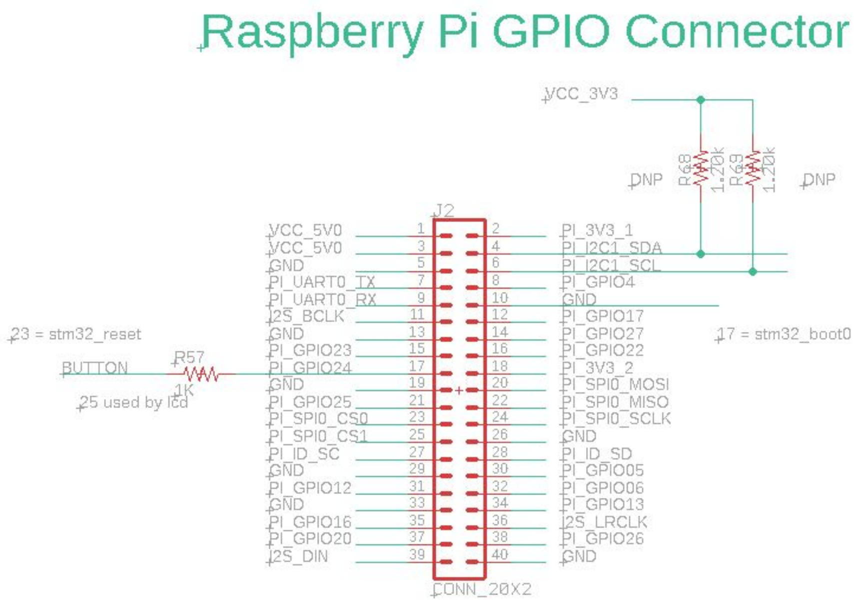
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Wombat Developer Manual

Hardware Overview

Pinout Table for Wombat J2 (Pi)





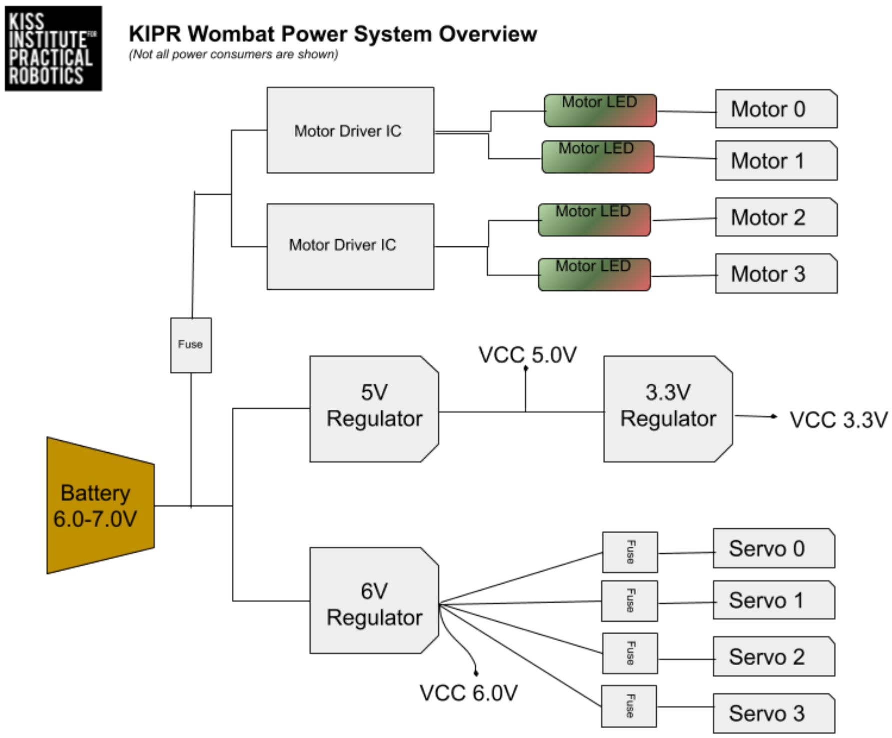
Keep In Mind: The GPIO number does not necessarily match the J2 Pin Numbering (some aren't GPIO)

Pin	Description
1,3	Sourced by the 5V Regulator; Powers the Raspberry Pi (indirectly from the battery)
2,8,14 ,16,18	Not Actually Connected to Anything; These are potential for new features or modifications
4,6	Serial EEProm (Serial Number for Wombat); Touch Screen Controller; Real Time Clock (DNP);
7,9	STM32 UART Communication; https://github.com/kipr/Wombat-Firmware for more information.
11	I2S line for audio circuit (DNP)
12	Controls the boot setting of STM32 (2 bit, Boot0 and Boot1 which is pulled low), see datasheet U37
15	STM_RESET, This will reset the STM32; For specifics on what that means, see datasheet U37
17	Takes the physical button input (GPIO set to input mode)
20,22, 23,24, 25	SPI Communication with the STM32. Despite having two select lines only the STM32 is connected, so it is possible to add more SPI devices in the future. Pin 25 (CS1) is not even connected at all.
26-35	Not Actually Connected to Anything; These are potential for new features or modifications
36,39	Intended for speaker circuit LRCLK and I2S_DIN but was removed during development (DNP)
35-38	Not Actually Connected to Anything; These are potential for new features or modifications

*DNP = Do Not Place, The component/circuit is **not** soldered on and the pads are empty in production*

Wombat Developer Manual

Power System Overview



VCC_5V0 (5 Volt Regulator Source)	Raw Battery
Raspberry PI (Via J2 Connector)	Battery Level Measuring Circuit (STM32)
3V3 Regulator (U7)	Power Switching Circuit
Servo Controller	No Fuse Battery
Speaker and Amplifier (DNP)	6V Regulator (U72)
HDMI Output (Pi's 5V source which comes from VCC_5V0)	5V Regulator (U73)
VCC_3V3 (3 Volt Regulator Source)	Fused Battery
LEDs (Red Power LED and Yellow Warning LED)	Motor Driver Circuits
Push Button (used for LOW signal)	VCC_6V0 (6 Volt Regulator Source)
Digital Ports (used for HIGH signal)	Servos (Fused)
Analog Ports (Used for HIGH signal)	
Real Time Clock and Coin Cell Charging (Not Used)	
Servo Controller	
Motor Back EMF Measurement	
I2C Registers	
Inertial Measurement Unit (IMU) (Accelerometer, Magnetometer, and Gyroscope)	