Extension Boards for the Wombat

Christian Kargl
HTL St. Pölten
St. Pölten, Austria
christian.kargl21@gmail.com

Peter Strommer
HTL St. Pölten
St. Pölten, Austria
p.strommer@outlook.at

Christoph Hackl
HTL St. Pölten
St. Pölten, Austria
christoph.hackl@htlstp.at

Manuel Bichl
HTL St. Pölten
St. Pölten, Austria
manuel.bichl@htlstp.at

***Abstract*—This paper presents the implementation and its uses of several extension boards for the Wombat. Our boards include a servo-shield and a gpio extender, which enhances the functionality and capability of the Wombat system. The servo-shield allows precise control of multiple servo motors, making it ideal for ECER. Meanwhile, the gpio extender expands the number of available input and output pins.**

Keywords—Servo-Shield

I. Introduction

The Wombat controller is broadly used at the competitions of ECER. With its sensor and servo ports it is capable of various tasks, but the number of ports may be limiting for more advanced ones. To address this limitation, teams at ECER can expand their Wombat controllers with additional shields and extender boards, enabling them to control more peripherals. This paper focuses on extensions that communicate via the I²C bus, specifically the Adafruit Servo Shield, which utilizes the PCA9685 microcontroller, and the GPIO Extender, which employs the PCF8574 chip.

II. Software Development

A. Suitable Libraries

The “Servo-Shield” and “Gpio-extender” both needed special libraries to control and manipulate. After careful consideration, the “Adafruit Servokit” and “Adafruit PCF8574“ libraries were chosen. These libraries provided easy-to-use, high-level APIs and simultaneously the flexibility that was required for this project. Usage of Libraries:

1. Adafruit Servokit

The Servokit library from Adafruit was able to easily communicate with the “PCA9685” microcontroller. This library is used to control an additional 16 servos and continuous servos via the I²C bus. The usage of the library in Python is as follows:

```
from adafruit_servokit import ServoKit
kit = ServoKit(channels=16)
```

After creating the servokit object and passing the appropriate number of channels as an argument, it is ready to use. The library can control both standard servos and continuous servos. The difference between the two is that standard servos operate in a specific range of motion, for example, 0–180 degrees. On the other hand, the continuous servos act more like conventional motors, meaning they can rotate freely.

1.1 Servos

```
kit.servo[0].angle = 0
```

This code would, if executed correctly, modify the signal sent to the servo connected on the shield's number 0 pin so that it moves to the desired angle.

1.2 Continous Servos

```
kit.continuous_servo[1].throttle=0
```

This code modifies the signal so that the continuous servo stops moving. The range of the value is between -1 and 1. Some continuous servos could require some fine-tuning by turning a potentiometer directly on the physical servo. But if that is not present, the PWM signal can be modified in the code to calibrate it. This will be explained in further detail later in the paper.

2. Adafruit PCF8574

This library is used to control the additional 8 gpio pins we added to the Wombat via the I²C bus. Using the library is quite intuitive since it is very similar to the native way of controlling gpio pins on the Raspberry Pi. Usage of the library in Python is as follows:

```
from adafruit_pcf8574 import PCF8574
import board
pcf = PCF8574(board.I2C(), address=0x27)
```

After the object has been created, it can be used like this to write to a specific pin:

```
write_pin(PIN, VALUE)
```

The PIN value can range anywhere from 0 to 7, and the value is either 0 or 1. Reading from the Extender is also quite easy.

```
pcf.read_pin(PIN)
```

Again, the PIN value can range from 0 to 7.

B. Calibrating Servos

Servo pulse timing varies between different brands and models. Due to the analog nature of the control circuit, there is often variation even between servos of the same brand and model. For precise position control, the minimum and maximum pulse widths must be calibrated in the code to match the known positions of the servo.

1) Finding the Minimum Pulse Width

```
kit.servo[0].set_pulse_width_range(SERVOMIN,
SERVOMAX)

kit.servo[0].angle = 0
```

Using the provided example code, the SERVOMIN value is adjusted until the low point of the servo's sweep reaches the minimum range of travel. It is recommended to approach this gradually and stop before the physical limit of travel is reached.

2) Finding the Maximum Pulse Width

Similarly, the SERVOMAX value is adjusted in the example code until the high point of the servo's sweep reaches the maximum range of travel. Again, this should be approached gradually to avoid reaching the physical limit of travel.

III. HARDWARE EXPLANATION

1. Servo board

Driving servo motors using the adafruit servokit library is considered straightforward, but each servo consumes a dedicated pin, along with some processing power from the Raspberry Pi. The Adafruit 16-Channel 12-bit PWM/Servo Driver is designed to control up to 16 servos over I2C using only 2 pins. The onboard PWM controller is capable of driving all 16 channels simultaneously without requiring additional processing overhead from the Microcontroller or Raspberry PI. Furthermore, up to 62 of these drivers can be chained together, allowing control of up to 992 servos, all using the same 2 pins.

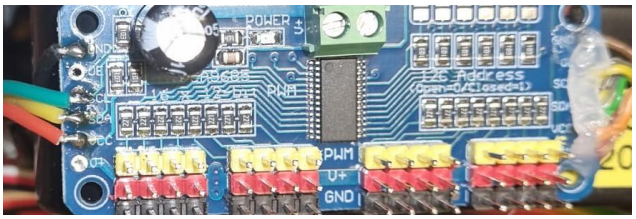

Fig. 1 Servo-Shield

Two sets of control input pins are located on either side of the board. Both sets of pins are identical, enabling multiple boards to be easily chained together by connecting them side by side.

Power Pins:

- GND: This pin is used for power and signal ground and must be connected.
- VCC: This pin supplies logic power and should be connected to the desired logic level for the PCA9685 output. Since the WOMBAT already uses the I²C pins, it is recommended to match the logic voltage of 3.3V.
- V+: This is an optional power pin that can be used to supply power to the servos. If servos are not being used, this pin can remain disconnected. It is not utilized by the chip itself. When used, an input of 5-6V DC is recommended, which can be sourced from the Raspberry Pi. However, as the supply current of the Raspberry Pi is limited, a power source external to the Wombat is recommended. Otherwise, the Wombat could be damaged, or the servos might not receive sufficient power to operate. Keep this in mind when using multiple "extender boards".

Control Pins:

- SCL: The I²C clock pin, which is connected to the microcontroller's I²C clock line. It can operate with 3V or 5V logic and includes a weak pull-up to VCC.
- SDA: The I²C data pin, which is connected to the microcontroller's I²C data line. It can also operate with 3V or 5V logic and includes a weak pull-up to VCC.
- OE: The output enable pin, which can be used to quickly disable all outputs. When this pin is set to low, all outputs are enabled. When set to high, the outputs are disabled. By default, it is pulled low, making it an optional pin.

Output Ports:

Sixteen output ports are provided, each consisting of three pins: V+, GND, and the PWM output. Each PWM output operates independently, but all must share the same PWM frequency. The maximum current per pin is limited to 25 mA.

Connecting to the Raspberry Pi

The PWM/Servo Driver is connected to the Raspberry Pi using I²C, requiring only five wires:

- 3.3V → VCC (this supplies power to the breakout board only, not the servos).
- GND → GND.
- 5V → V+ (while it is not ideal to use the Raspberry Pi's 5V directly, it is the simpler option).
- SDA → SDA.
- SCL → SCL.

Chaining Multiple Boards

If more than 16 servos need to be controlled, additional boards can be chained together. With headers present on both sides of the boards, the wiring process is simplified, requiring only a 5-pin parallel cable to connect one board to the next.

Addressing the Boards

Each board in the chain must be assigned a unique address. This is accomplished using the address jumpers located on the upper right edge of the board. The base I²C address for each board is 0x40. The binary address, which is programmed using the address jumpers, is added to the base I²C address. To program the address offset, a drop of solder is used to bridge the corresponding address jumper for each binary '1' in the address.

- Board 0: Address = 0x40, Offset = binary 00000 (no jumpers required).
- Board 1: Address = 0x41, Offset = binary 00001 (bridge A0).
- Board 2: Address = 0x42, Offset = binary 00010 (bridge A1).
- Board 3: Address = 0x43, Offset = binary 00011 (bridge A0 & A1).
- Board 4: Address = 0x44, Offset = binary 00100 (bridge A2).

2. Gpio expander

The idea to implement a gpio expander came from the servo expander board. After noticing that no motor pins were free and that a MOSFET still needed to be powered, the conclusion was to simply add another board—this time, a gpio expander board.

After choosing the PCF8574 gpio expander board, the installation went smoothly, except for the unnoticed pull-up resistors on the SDA and SCL lines. After identifying the issue and desoldering them, the gpio expander worked, and we had another 8 gpio Pins.

Again, there were control inputs on both sides of the board, making chaining them quite easy. Just make sure to desolder the pull-up resistors; otherwise, issues may arise.
You can chain up to 8 expanders with the 3 jumpers on the board for changing the address and having a total of 64 gpio Pins added.

Power Pins:

For the power pins, it is the same as the servo expander board, except that there is no V+. Therefore, keep in mind that all devices connected to or powered by the board operate at 3.3V. This may cause issues with sensors that operate at 5V. Other than this change, you can simply just chain it to the servo extender port.

How it works:

This chip does not have a pin direction register. You cannot set the pins to be input or output - instead each pin has two possible states. Basically, you can think of it as an open-drain output with a 100K resistor pull-up built in.

- Option one: Lightly pulled up 'input' - by default it will read as a high logic level, but connecting the gpio to ground will cause it to read as a low logic level.

- Option two: Strong 20mA low-driving transistor sink output. This means the output is 'forced' to be low and will always read as a low logic level.

The pin direction / state thing is a little odd but it actually works fine for many purposes as long as you know what to expect.
For example, if you want to read a button or switch, connect one side to the PCF and the other side to ground. Then set the pin to 'light pull-up input'. When the button is pressed it will read low, when released it will read high.
If you want to control an LED, connect the anode to positive voltage through a resistor. When the PCF pin is set to 'light pull-up input' the LED will be off. When the PCF pin is set to 'strong ground output' the LED will connect to ground and turn on.
If you want to send a gpio output logic level to some other device or peripheral, the light pull-up acts as high logic out, the strong ground output acts as low logic out.
If you want to receive a gpio input logic level, set the pin to light pull-up and then read the pin to determine if the gpio input is high or low.
Basically, the only thing to watch for is you cannot drive an LED that is expecting the expander gpio to go high to turn on the LED, or connect a button input to a positive voltage without adding an additional pull-down resistor.

Since this is very confusing when hearing it for the first time, you shouldn't worry too much since most of it is taken care of in the Adafruit PCF8574 library - you can pretend it has input/output modes and the library will fake out what you are expecting.

IV. REFERENCES

[1] Adafruit Servo Kit:
https://learn.adafruit.com/adafruit-16-channel-servo-driver-with-raspberry-pi/using-the-adafruit-library

[2] Adafruit 16-Channel 12-bit PWM/Servo Driver
https://www.adafruit.com/product/815

[3] Adafruit PCF8574 I2C GPIO Expander
https://www.adafruit.com/product/5545

[4] Adafruit PCF8574 I2C GPIO Expander Arduino library
https://github.com/adafruit/Adafruit_PCF8574

[5] Adafruit PCF8574 I2C GPIO Expander python library
https://learn.adafruit.com/adafruit-pcf8574/python-circuitpython