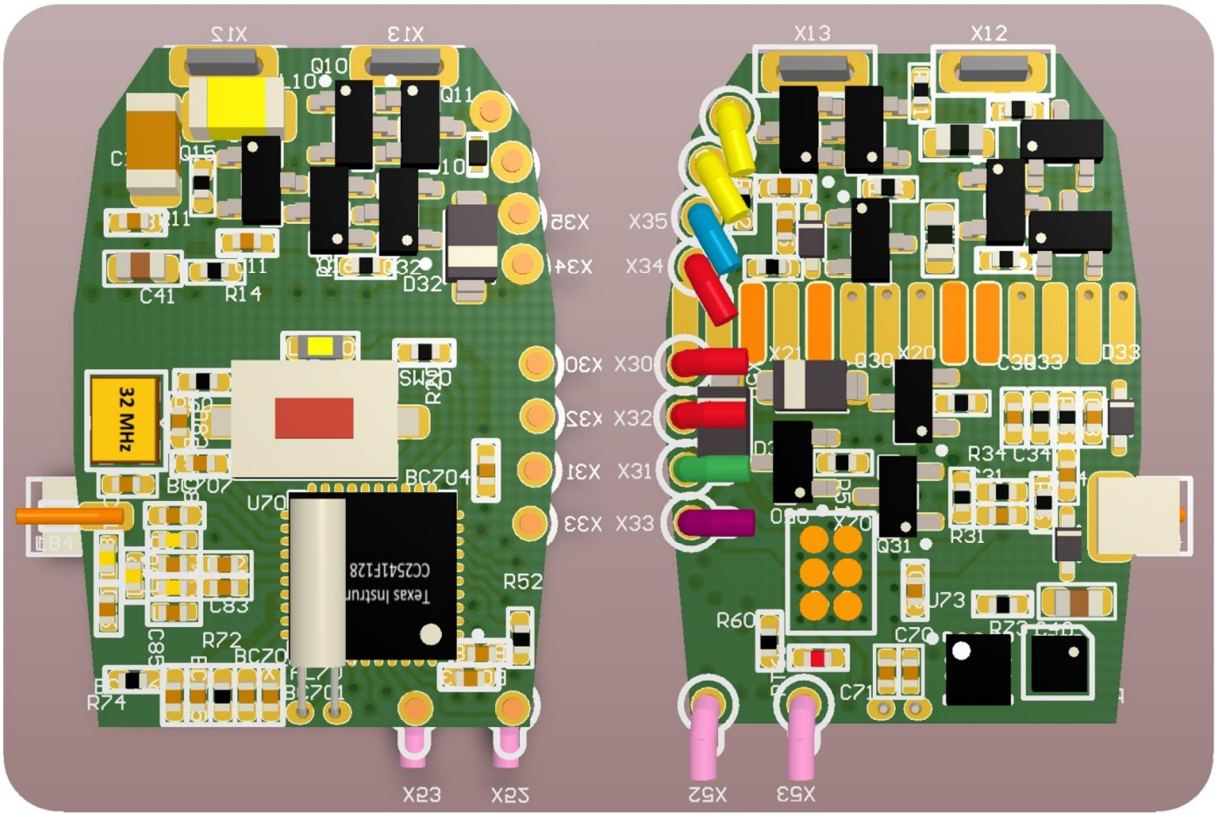
WeVibe 5: HW/FW Platform PCB

Design Document

Revision 18
(In Progress)



- - Confidential - -

Modes of Operation

Pre-Programmed Modes

The proposed modes of operation of the WeVibe 5 are as shown below:

Mode	Name		Description
1	Low		Both motors constantly ON, low intensity
2	Medium		Both motors constantly ON, medium intensity
3	High		Both motors constantly ON, high intensity
4	Thrust		C motor rate is dependent on rate of male thrusting, information acquired via accelerometer.
5	Pulse		Both motors ON at set intensity then OFF, approximately 2Hz, 50% duty cycle
6	Echo		Primary ON at set intensity while Secondary is OFF, then alternate at about 2 Hz.
7	Wave		Both motors ramp from low intensity, to set intensity, and back again.
8	Tide	Inverted Wave	Primary ramps from low, to set intensity, while Secondary ramps from high to low.
9	Programm-able		If enabled via the app, this mode is accessible via the button on the unit.

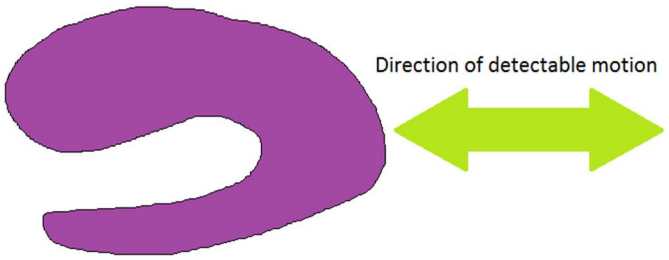
Main WeVibe 5 Device

Overall, the main device operates the same as the WeVibe 4+, except with new firmware to support the additional modes of operation (based on an accelerometer).

Sensor Support

Accelerometer

An accelerometer is included on the main PCB to react to the movements of the WeVibe. Presently, the only reaction we do is shown in the “Thrust” mode.



In this mode, the speed of the motors are dependent on the back-and-forth movement of the vibrator. This is expected to be mostly due to the thrusting of the male during sexual activity.

Pressure Sensor

One use that is possible is a pressure sensor in the tip of the WeVibe that senses the pressure in the vagina. As the male thrusts in and out, the pressure will increase when he is in, and decrease when he is out.

Circuit Description, Hardware Revision

The sections of the schematic are divided by function, and annotated using the reference designators. For example, the motor driver is section 3 as indicated by the reference designators Q30, D32, R36, etc; all designators begin with a "3".

Section 1: Battery Charging

The non-polarized power is applied to the charger pins at X10 & X11 and is rectified using the active bridge Q10-Q13. Input transients are absorbed by D10. If power is supplied via inductive charging, C10 is used as the compensation capacitor.

If port P0.7 is toggled, the voltage at the gate of Q17 remains high, thanks to capacitor C15 and diode D11. R102 is used to ensure the FET is always OFF when no toggling occurs. With Q17 turned ON, the gates of Q14 and Q15 are pulled low, turning these ON. When toggling ceases, resistor R11 ensures the FET's turn OFF quickly so as to not waste energy. Q15 is the main power switch and Q14 is used as a switchable diode. With charging enabled, the diode is short-circuited by the closed channel and when disabled, the diode ensures that battery voltage does not appear at the Vin node, falsely indicating that a charger is attached.

Resistor R10 and R101 ensure the default state of Q16 is at ground. R10 also allows port P0.7 to double as a charger detect pin. When not charging, port P0.7 also controls Q16 which is used to enable the battery voltage measurement divider of R13/R14, allowing a reduced version of the battery voltage to be read via port P0.5. The switch Q16 is needed to allow a low-impedance battery reading while still ensuring low sleep current. The switch is placed in the middle of the ladder rather at ground since if the source was at GND and the bottom of R14 was on the drain, port P0.5 would have a low-resistance path to Vbatt which would draw current when the battery read function is disabled. This switch also allows this resistor ladder to be used as a current dump in the case that charging is finished and the nominal charge current is too much.

Resistor R12 allows this afore-mentioned nominal charge current to flow into the battery from the attached charger, regardless of the processor state. If a battery is really dead, so dead that the processor cannot run, it must be possible to charge the battery. With a charger attached to a dead device, voltage is dropped on the body diode of Q14 then current flows through R12, L10, into the battery. This current into a battery at 1.0 volts is about 20 mA, and into a 4.2 volt battery is 2.7 mA. This is for a 620 mAh battery; other capacities have similarly proportional values. Once a battery is fully charged

Filter C12/L10 cleans up the charge current pulse created by the PWM which makes reading the current more accurate. Pulse current charging a lithium battery is actually a good thing so the filter doesn't have to be perfect. The charge current measurement is taken by the processor across R15, and this signal is filtered a bit by C13. I say it is filtered "a bit" since the resistor is so low in value that the cut-off frequency of the filter is over 700 kHz. The other purpose of C13 is to allow any high current transients to bypass the resistor. If a 2Amp transient had to pass through the resistor, it would cause a 440 mV jump in the ground potential difference between the motor ground and system ground. This current

measurement system has a resolution of 350 µA per count of the 14-bit ADC. The current limit required for the topping charge of the battery is 18.6 mA resulting in the limit being set to 53 counts of the ADC. Even this is more accurate than required; perhaps I can reduce the sense resistor to 0.10 Ω for even better transient performance.

Our previous algorithm simply charged the battery to 4.2 volts at a fairly constant current. However, this new circuit allows the implementation of constant current or constant voltage (CC or CV) regulation, thereby enabling the possibility of a second phase, constant voltage charge to bring the battery level right to 100%. By not performing this last step when charging at a high rate, we would sacrifice about 30% of battery charge. In previous designs, the overall charge rate was so low that the CC charge would bring the cell to over 90% without the need for a topping charge. Since lithium batteries don't particularly like being driven to 4.2 volts for extended periods, we decided to sacrifice a bit of run time in exchange for a longer life battery.

One problem with Classic was the inability to easily change the charge balancing between the nominal charging current and the current required for operation. Once charging is complete, we must stay awake in order to keep the LED turned ON. We don't have to stay awake, but we can. Even so, the LED itself will draw current, as will the processor, both draining the battery. At the same time, the nominal charge current might be 6 mA at a fully charged battery. These two currents must combine to result in a nominal drain on the battery as low as possible without the chance of charging the battery.

In this design, we have ultimate control over the charge rate, as low as the nominal charge rate. If we want a bit more charge, we can turn charging ON at a low duty cycle, or just turn it ON now and again. We are also able to read the current in/out of the battery (excluding motor current). This allows us to modify our operation as required. One problem is that the charge control pin cannot be toggled in order to PWM the battery measurement ladder since this operation will enable charging. The pin can however be set high to draw an extra 5 mA if required. Also, port P1.3 can be set high to draw an extra 3 mA if needed. The code will ensure the battery is not trickle-charged.

Section 2: Pushbutton Switch

As shown in the main board schematic, the processor will read zero volts when pushbutton SW20 is pressed, and due to R20, reads Vcc when it is open. Too simple… let's complicate it.

If the switch is not populated, ports X20 and X21 can be used to mount a switch somewhere else, allowing the switch position to be separated from the board location to allow more options in the mechanical design. That external switch connection can also be an array of switches, each shorting the two terminals with a different resistance. This varying voltage drop is measured by the processor allowing up to 8 switches to be distinguished.

Section 3: Motor Drivers

All three motors are driven in the same manner. The port pin drives the MOSFET high to turn the corresponding motor ON. The pull-down resistor on each channel ensures the motor will be OFF if the processor dies due to a really low battery. The diode is used to absorb any back EMF generated when the MOSFET switch opens.

Motors 1 and 2 are also able to have their rotational speed measured by the processor. Each channel passes through an RC filter and is read by the processor. During the OFF time of the PWM, the motor continues to rotate, operating like a generator. These voltage pulses are DC coupled (high pass filter) by C30 or C31, filtered (low pass) via R33/C33 or R34/C34, delivering a signal to the processor pin P0.1 or P0.0.

The diode in this circuit ensures that the negative-going pulse (caused by the DC blocking capacitor) does not allow too much current to flow through the processor's internal protection diode. The RB521 will kick in before the built-in diode, but really there isn't a lot of current capability here. I may not need the diode at all.

Section 4: Voltage "Regulator"

Since the processor can withstand a maximum of 3.6 volts on Vcc, but the battery may extend up to 4.2 volts, a voltage regulator is required. As we are always fighting a battle of cost, we use the body diode of a couple MOSFET's to drop the voltage as required.

R40 combine with capacitors C40 and C41 to form a filter, cleaning up the main Vcc supply from the battery. MOSFET Q40 is always disabled, so the body diode drop is always in play, approximately 0.4 volts. This brings a fully charged battery at 4.2 volts down to 3.8 volts. The processor monitors the battery voltage and at the proper levels, will engage or disengage Q41 to increase or decrease the Vcc level respectively.

Immediately after a change to the voltage drop, no analog measurements are made since the supply voltage may be in flux. This is not true of analog readings of the motor speed, but this is irrelevant since when a motor is running, the battery voltage drops significantly and the extra drop is never required.

After a lithium cell is fully charged and charging stops, the voltage drops to about 3.8 volts within a short time. This is important since the extra voltage drop cannot be engaged while the device is sleeping, lest we draw an extra 55 µA. The only time this can occur is when a charging cycle is finished, and immediately the device is removed from the charger, and prior to being charged, the device was paired to a remote control. If it was paired to a phone, the system stays awake after charging.

Regardless, prior to going to sleep, the battery voltage is monitored (and a light load is applied) until the voltage drops to a level such that the extra voltage drop can be eliminated. Nighty-night.

The end result is a "regulator" that serves the purpose from an electrical point of view, costs 3 cents, and consumes zero quiescent current. The only downside is the extra 9 mm^2 of board space required.

Section 5: LED's

All LED's are powered directly from the battery voltage Vbatt. A problem however is that if the diodes were connected directly to Vbatt, when disabled, there would be a low-impedance path from the battery to Vcc through the processor protection diode. To circumvent this, Q50 is used to turn the supply voltage ON and OFF for the LED's. Resistor R51 ensures that when not in use, the switch is turned OFF and the LED's won't draw any current.

Solder pads X53 and X54 are used to connect an external LED. The on-board resistor offers the option of using simple through-hole LED's without having to add the resistor inline as well. The intent is an external PCB with surface mount LED and surface mount resistor.

Sadly though, this resistor R51 will cause about 150 nA of current to flow (worst case). This shouldn't be bad news except the system presently draws about 1.8 µA when sleeping. This pesky resistor adds almost 5%. Regardless, a sleep current of 2 µA will allow a device that has just been run dead to sleep for over 1.5 years, and a ½ full device to sleep for 8 years.

Section 6: Thermistor

The thermistor is controlled by the processor port P2.1 so that we may save power during sleep mode. When enabled, the thermistor temperature is read by the processor at port P0.3. As the temperature rises, the value read falls. This inverse relationship isn't a problem since we simply look at particular trip points for operation and for charging.

Section 7: Main Processor and Sensors

Programming of the processor is accomplished via X70, a set of pads on the PCB. An external 2.7 kΩ resistor must be placed between the reset line and the programmer, built into the test jig.

XTAL70 is the sleep crystal that is installed on products that require extended operation with the BLE enabled. Otherwise it is not installed in order to save a few cents. R72 is a bias resistor required for the internal voltage reference. Integrated circuits U71 and U73 are the accelerometer and pressure sensor respectively. The I2C bus requires pull-up resistors R73 and R74 for proper operation but I always wonder about the pull-up on the clock line as the device is the bus master. Oh. Maybe if the I2C bus has the device configured as the slave, then the pull-up is needed for operation.

As both sensors have open-collector outputs on the IRQ line, we simply wire-OR them together to alert the processor on various pressure or accelerometer events.

Section 8: Clock and RF

Capacitors C80 and C81 apply the needed load to XTAL80, generating the main system clock. C82 and C83 are DC blocking capacitors at the processor RF output pins. A balun is created with L80/C84/L81/C85 and this now single-ended signal is matched to the antenna with L82/L83/C86. The antenna is a simple ¼λ wire whip antenna.