

PURRCEPTRON

ECE 445 DESIGN DOCUMENT - FALL 2026

Project # 34

Megna Naidu, Neel Rajesh, Vinny Smith

TA: Denghan Xiong

Contents

1	Introduction	4
1.1	Problem	4
1.2	Solution	4
1.3	Visual Aid	5
1.4	High Level Requirements.....	6
2	Design	7
2.1	Physical Design.....	7
2.2	Block Diagram	8
2.3	Functional Overview & Block Diagram Requirements	9
2.3.1	Power Subsystem	9
2.3.2	Control Subsystem	10
2.3.3	Sensing Subsystem.....	11
2.3.4	Expression Subsystem.....	12
2.3.5	Mechanical Subsystem.....	13
2.4	Hardware Design.....	14
2.4.1	Operating Voltage & Regulation.....	14
2.4.2	Battery Charging	15
2.5	Software Design	16
2.5.1	Touch Classification & Mood.....	16
2.5.2	Touch Sensing.....	18
2.5.3	Saving the Pet's State.....	20
2.6	Commercial Component Selection.....	20
2.6.1	Drive Motors	20
2.6.2	LED Matrix Face.....	20
2.6.3	Battery	21
2.7	Tolerance Analysis.....	21
2.7.1	Battery Voltage Under Load	21

2.8	Cost Analysis.....	22
2.9	Schedule	24
2.10	Risk Analysis.....	24
3	Ethics and Safety	26

1 Introduction

1.1 Problem

Loneliness and social isolation are a growing public health concern. Health researchers, including the U.S. Surgeon General, have linked long-term loneliness to a higher risk of heart disease, depression, and earlier death [1]. Spending time with animals is one way people cope with stress, anxiety, and loneliness, and emotional support animals are common for this reason. However, many people cannot own a pet. Housing rules, allergies, cost, travel, and the daily time needed for care all get in the way. Current alternatives do not solve this problem well. Virtual pet apps live on a screen, so there is nothing to hold and nothing that exists when the app is closed. Most consumer robot pets follow scripted behaviors that become predictable after a few uses, so users never feel a relationship developing. Robots made for care settings, such as the PARO seal used in dementia care, respond to touch and have shown promising results in studies [2]. However, they are expensive and are built mainly for institutions, not for individual owners.

1.2 Solution

To help with this, we will build Purrception, a low-cost, hand-sized robot cat that reacts to how it is touched and remembers how it has been treated. Eight copper tape touch pads inside a 3D-printed shell and a motion sensor will let the pet tell apart a tap, a pet (a stroke along its back), a hold, and being picked up. It will answer with animated eyes on an LED matrix, chirps from a small speaker, a wagging tail, and short drives on two wheels.

Inside, the pet keeps track of its mood, energy, and attachment. These values change with every interaction and slowly drift over time. A real-time clock keeps running while the pet is off, so when it turns back on it knows how long it was alone. This means the same stroke gets a happier response from a pet that is handled every day than from one that has been ignored. The pet has no camera, microphone, or network connection, so no user data ever leaves the device.

1.3 Visual Aid

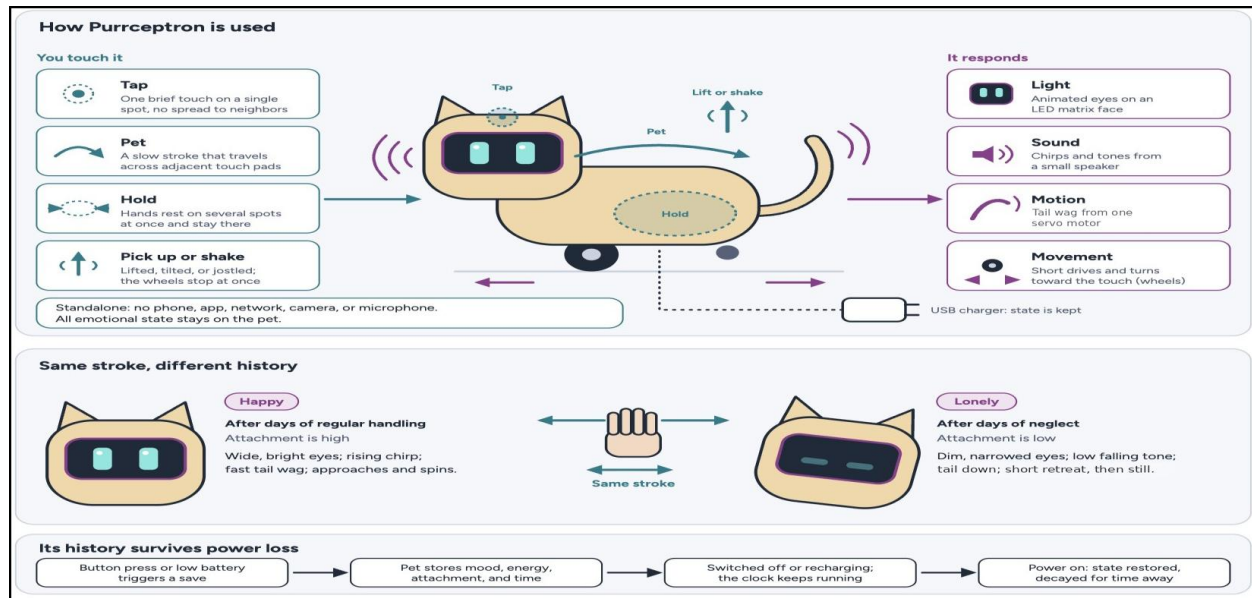


Figure 1: How Purrceptron is used

Figure 1 shows how a person uses Purrceptron. The user touches the pet in one of four ways: a tap, a pet, a hold, or picking it up. The pet answers with light, sound, tail motion, and movement, and how happy it acts depends on how it has been treated over the past hours and days.

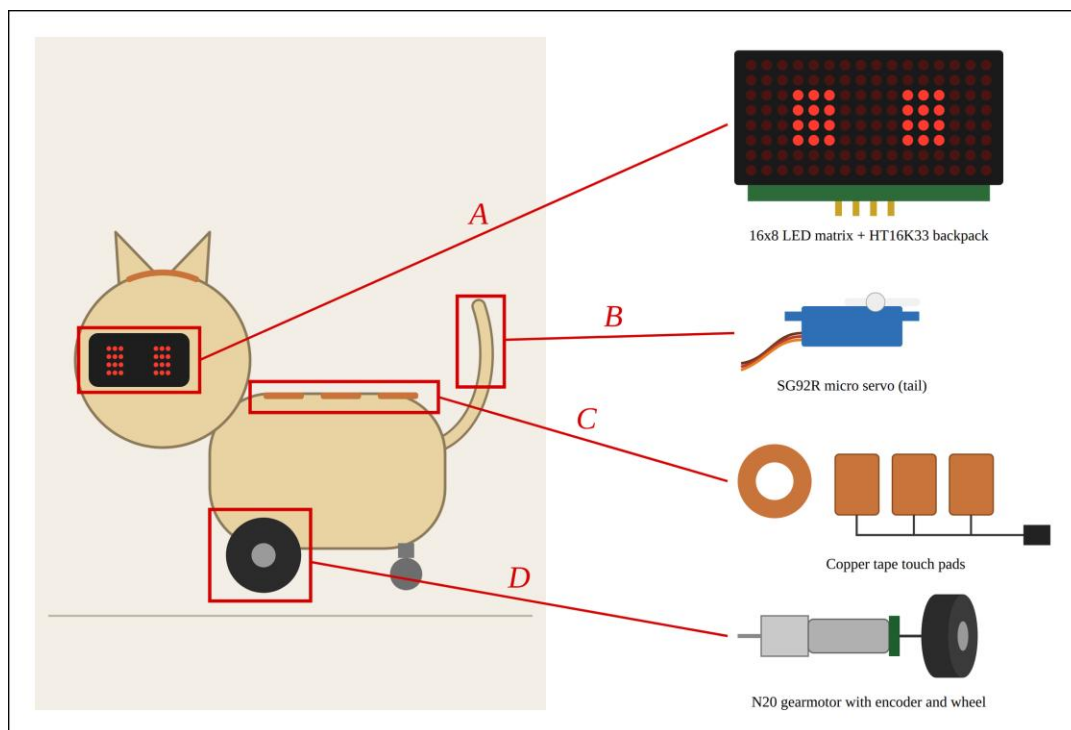


Figure 2: Visual aid of the main parts of Purrceptron

The face of the pet is a 16x8 LED matrix, as seen on label A in Figure 2. It shows animated eyes, and it is driven over I2C by the driver chip on its backpack board.

At the back of the pet, a small servo motor wags the tail. This is seen as label B on Figure 2.

The shell of the pet is 3D printed, and eight copper tape touch pads are stuck to the inside of the wall. This is seen as label C on Figure 2. When a finger touches the outside of the shell over a pad, our microcontroller sees the change in capacitance.

To move around, the pet uses two small N20 gearmotors with encoders, plus a ball caster in the rear. This is seen as label D on Figure 2. All of these parts connect to one custom PCB inside the body, which holds the ESP32-S3 microcontroller, the battery charger, the power regulators, the motion sensor, the speaker amplifier, and the motor driver.

1.4 High Level Requirements

To consider our project successful, Purrceptron must fulfill the following:

1. Purrceptron is able to tell apart a tap, a pet, a hold, and being picked up, and classifies each one correctly in at least 18 of 20 trials. Its first light or sound reaction starts within 100 ms of the start of a touch.
2. Purrceptron saves its mood and energy whenever it is turned off with the power button or when its battery gets low. After being off for 30 minutes to 24 hours, it restores these values to within 5% of the expected values.
3. Purrceptron shows three clearly different moods (happy, neutral, and lonely) using its face, sounds, tail, and movement. People who have not seen the pet before identify its mood correctly at least 80% of the time.

2 Design

2.1 Physical Design

Figure 3 shows how the parts sit inside the pet. The shell is printed in two halves that screw together, so we can open it to charge, program, or fix the pet. The LED face sits behind the front of the head, and the speaker sits in the front of the body. The PCB is mounted on standoffs in the middle of the body, with the battery under it to keep the weight low. The two drive wheels share one axis near the front, and a ball caster carries the rear. The tail servo is screwed to the inside of the shell at the back. The whole pet fits inside a 15 cm x 12 cm x 10 cm box.

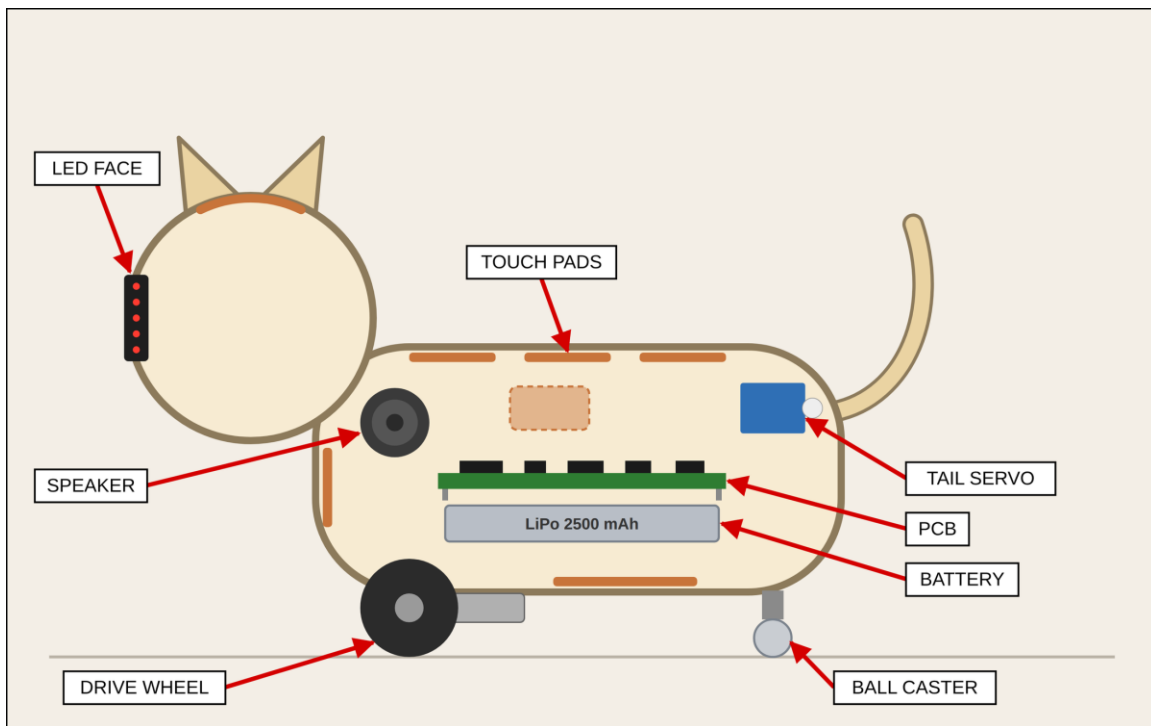


Figure 3: Layout of the parts inside the shell

For touch sensing, we will stick copper tape pads to the inside of the shell wall. There is one pad on the head, three in a row along the back, one on each side, one on the chest, and one on the belly. Each pad is wired back to a pin header on the PCB. When a finger is on the shell, as seen in Figure 4 on the left, the finger and the pad act like the two plates of a small capacitor through the 2 mm plastic wall, which raises the capacitance of the pad. When no one is touching the shell, as seen in Figure 4 on the right, the pad only has its normal capacitance to ground.

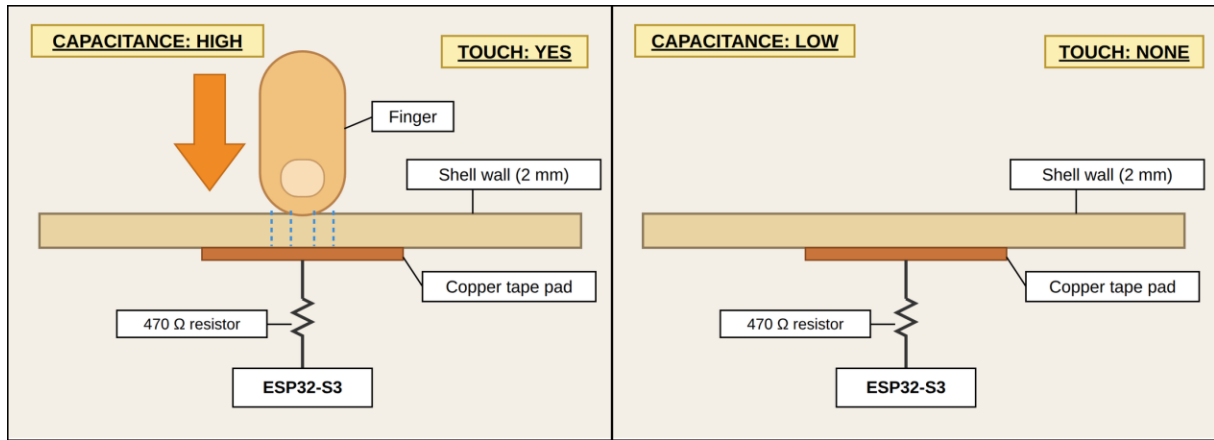


Figure 4: Touch sensing through the shell wall

2.2 Block Diagram

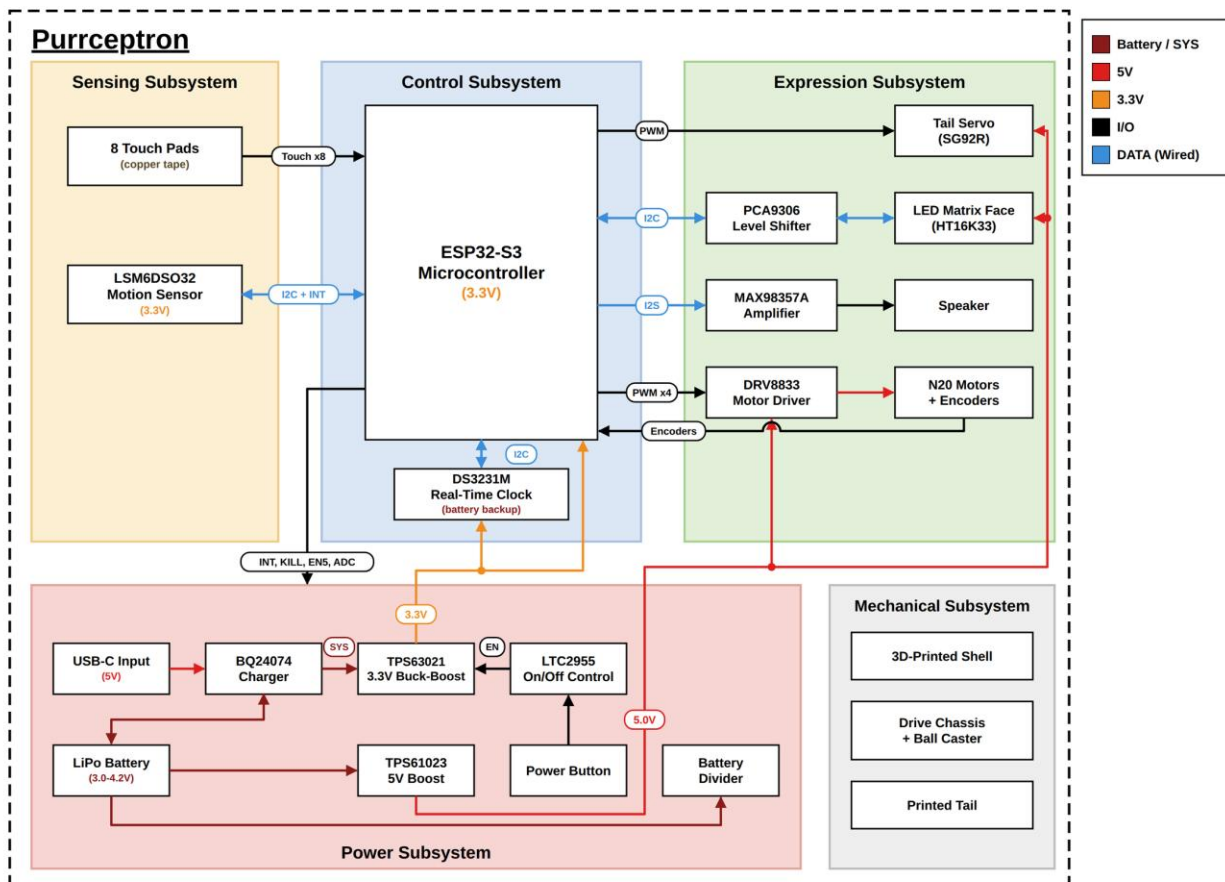


Figure 5: Purrcpeptron Block Diagram

2.3 Functional Overview & Block Diagram Requirements

2.3.1 Power Subsystem

The Power Subsystem is responsible for supplying power to every other part of the pet. It charges a single 3.7 V lithium polymer battery from a USB-C port using a BQ24074 charger, which also lets the pet run while it charges. A TPS63021 buck-boost converter makes the 3.3 V rail for the microcontroller and the sensors, and a TPS61023 boost converter makes the 5 V rail for the LED face, the tail servo, and the motor driver. An LTC2955 pushbutton controller turns the pet on and off, and it lets the microcontroller save the pet's state before power is cut. A resistor divider lets the microcontroller read the battery voltage. For more information on the hardware design of the Power Subsystem, please refer to Section 2.4. To ensure that the Power Subsystem is fulfilling its responsibilities, a requirements & verification table can be found below.

Table 1: Power Subsystem - Requirements & Verification

Requirements	Verification
The 3.3 V rail must stay within 3.3 V $\pm$ 0.1 V for any battery voltage from 3.0 V to 4.2 V, with up to 300 mA of load.	- Replace the battery with a bench power supply connected to the battery connector, and set it to 4.2 V. - Turn the pet on with the power button. Connect an 11 Ω power resistor from the 3.3 V test point to ground to draw about 300 mA. - Measure the 3.3 V test point with a multimeter with and without the resistor. Repeat with the supply at 3.7 V and 3.0 V. Confirm that every reading is between 3.2 V and 3.4 V.
The 5 V rail must stay within 4.95 V $\pm$ 0.25 V with up to 0.9 A of load while EN5 is high, and must turn off when EN5 is low.	- With the bench supply at 3.7 V, load a build of the firmware that sets EN5 high. Connect a 5.6 Ω power resistor from the 5 V test point to ground (about 0.9 A). - Measure the 5 V test point with a multimeter at supply voltages of 4.2 V, 3.7 V, and 3.0 V. Confirm the readings are between 4.70 V and 5.20 V. - Then, set EN5 low and confirm that the 5 V test point reads below 0.1 V.
A press of the power button while the pet is on must never cut power by itself. The pet must only turn off after the microcontroller saves its state and pulls the KILL line low.	- Load a build of the firmware that waits 2 seconds after a button press before pulling KILL low. - Connect an oscilloscope to the PWR_INT line, the PWR_KILL line, and the 3.3 V test point. Then, press the power button. - Confirm that the 3.3 V rail stays on for the full 2 seconds and turns off within 50 ms after KILL goes low.

Requirements	Verification
The battery must be able to recharge from a 5 V USB-C port, and charging must stop automatically when the battery is full.	- Start with a battery that reads about 3.7 V. Plug a USB-C cable from a laptop into the pet and confirm the red charging LED turns on. - Measure the charge current with a multimeter in series with the battery. Confirm it is between 0.35 A and 0.5 A. - Stay with the pet until the red LED turns off. Confirm the battery voltage is $4.2\text{ V} \pm 0.05\text{ V}$ and no current is flowing into the battery.
The microcontroller must read the battery voltage to within $\pm 4\%$ of the true value.	- Connect the bench supply to the battery connector. Step the voltage from 3.0 V to 4.2 V in 0.2 V steps. - At each step, record the battery voltage read by the microcontroller via serial debugging and the voltage read by a multimeter. Confirm the two values are within 4% of each other.

2.3.2 Control Subsystem

The Control Subsystem is the brain of the pet. It is made up of an ESP32-S3-WROOM-1 microcontroller module and a DS3231M real-time clock. The microcontroller reads the touch pads and the motion sensor, decides which interaction happened, updates the pet's mood, energy, and attachment, and picks an expression. It talks to the motion sensor, the clock, and the LED face over I2C, sends audio to the speaker amplifier over I2S, and drives the tail servo and the motor driver with PWM. The real-time clock runs from the battery, so it keeps time while the pet is off, which lets the pet know how long it was alone. The microcontroller saves the pet's state into its flash memory. For more information on the software design of the Control Subsystem, please refer to Section 2.5. To ensure that the Control Subsystem is fulfilling its responsibilities, a requirements & verification table can be found below.

Table 2: Control Subsystem - Requirements & Verification

Requirements	Verification
The Control Subsystem must start the pet's first light or sound reaction within 100 ms of the start of a touch.	- Wire a push button from touch pad 1 to ground, so that pressing it acts like a finger touching the pad. - Load a build of the firmware that sets a spare pin high when it sends the first face frame. Connect an oscilloscope to the touch pad line and to the spare pin. - Press the button 20 times. Confirm that each time, the spare pin goes high within 100 ms of the button press.

Requirements	Verification
When the power button is pressed or the battery reading drops to 3.6 V, the Control Subsystem must save the pet's state to flash in under 200 ms before the pet turns off.	- Load a build of the firmware that holds a spare pin high while it is saving. Connect an oscilloscope to the spare pin. - Press the power button and confirm that the save takes less than 200 ms. - Then, replace the battery with a bench supply set to 3.8 V and slowly lower it. Confirm that the save starts when the supply reaches about 3.6 V.
When the pet is turned back on, the Control Subsystem must restore mood and energy to within 5% of the values expected from the time it was off.	- Set the mood to 80 and the energy to 40 with a serial debugging command, then turn the pet off. - Turn the pet on after 30 minutes. Record the mood and energy read via serial debugging and confirm they are within 5% of the values from our decay spreadsheet. - Repeat this test, but move the real-time clock forward by 2 hours and by 24 hours with a serial debugging command before turning the pet on.
When the motion sensor detects that the pet was picked up, the Control Subsystem must stop the drive motors within 200 ms.	- Command the pet to drive forward slowly on a table. - Lift the pet off of the table. Using an oscilloscope on the motion sensor interrupt line and on one encoder line, confirm that the wheels stop turning within 200 ms. - Repeat this test 20 times.

2.3.3 Sensing Subsystem

The Sensing Subsystem is responsible for telling the microcontroller how the pet is being handled. It is made up of eight copper tape touch pads and an LSM6DSO32 motion sensor. Each touch pad is wired through a 470 Ω resistor to one of the ESP32-S3's built-in capacitive touch pins, so we do not need a separate touch sensor chip. The motion sensor measures acceleration and rotation, and it sends an interrupt to the microcontroller when the pet is lifted or shaken. For more information regarding the physical design of this subsystem, refer to Section 2.1, and for the software design of the touch sensing, refer to Section 2.5.2. To ensure that the Sensing Subsystem is fulfilling its responsibilities, a requirements & verification table can be found below.

Table 3: Sensing Subsystem - Requirements & Verification

Requirements	Verification
Each of the eight touch pads must detect a fingertip touch through the shell in at least 19 of 20 touches.	- Ensure the shell is fully assembled and the pet is running on battery power. - Touch each pad 20 times for about half a second. Record the touch events read from the microcontroller via serial debugging. - Confirm that each pad registered at least 19 of the 20 touches.
The touch pads must not register more than one false touch in 5 minutes while the speaker plays and the motors run.	- Place the pet on a table and load a build of the firmware that plays sounds and drives back and forth. - Without touching the pet, record the touch events read via serial debugging for 5 minutes. Confirm there is at most one.
The motion sensor must send an interrupt within 100 ms of the pet being lifted, and must not send one during normal driving.	- Place a small lever switch under the pet so that it opens when the pet leaves the table. Connect an oscilloscope to the switch and to the motion sensor interrupt line. - Lift the pet 20 times. Confirm that the interrupt goes high within 100 ms of the switch opening in at least 19 of the 20 lifts. - Then, drive the pet around for 2 minutes and confirm that no interrupts are recorded via serial debugging.

2.3.4 Expression Subsystem

The Expression Subsystem is responsible for showing the pet's mood through light, sound, tail motion, and movement. A 16x8 LED matrix with an HT16K33 driver on its backpack board shows animated eyes. Since the HT16K33 runs at 5 V and our I2C bus runs at 3.3 V, a PCA9306 level shifter connects the two. A MAX98357A amplifier receives audio from the microcontroller over I2S and drives a small 4 Ω speaker. An SG92R micro servo wags the tail, and a DRV8833 motor driver runs the two N20 drive motors. The LED face, servo, and motor driver run from the 5 V rail, and the amplifier runs from the battery side of the charger. To ensure that the Expression Subsystem is fulfilling its responsibilities, a requirements & verification table can be found below.

Table 4: Expression Subsystem - Requirements & Verification

Requirements	Verification
The LED face must show at least three different eye shapes and update at least 10 times per second.	- Load a build of the firmware that cycles through the happy, neutral, and lonely eye animations. - Record the number of frames sent each second as read via serial debugging. Confirm that it is at least 10, and confirm by looking at the face that the three eye shapes are different.

Requirements	Verification
The speaker must play tones loud enough to hear across a room, at least 60 dB measured 30 cm away.	- In a quiet room, play 200 Hz, 1 kHz, and 4 kHz tones at the firmware volume limit. - Hold a phone sound level meter app 30 cm from the pet. Confirm that each tone reads at least 60 dB.
The tail servo must move between its two end positions when commanded, and must never move while the drive motors are running.	- Command the tail to its left and right end positions and confirm that it reaches both. - Load the full behavior firmware and let it run for 10 minutes. Using an oscilloscope on the servo signal and on the motor sleep line, confirm that the servo never receives pulses while the motors are on.
The pet must drive at 0.05 to 0.2 m/s and drive a 1 m straight line with less than 10 cm of drift.	- Mark a 1 m line on the floor. Command the pet to drive straight along it 10 times, timing each run with a stopwatch. - Confirm that the speed is between 0.05 and 0.2 m/s, and that the pet ends up less than 10 cm from the line in at least 9 of the 10 runs.

2.3.5 Mechanical Subsystem

The Mechanical Subsystem is responsible for holding all of the parts and keeping the pet safe to handle. It is made up of the 3D-printed shell, the drive chassis that holds the motors, wheels, and ball caster, and the printed tail. It has no electrical connections, but it affects the other subsystems directly. The wall thickness over the touch pads changes how well touch is sensed, and the alignment of the motors changes how straight the pet drives. For more information on the physical design, refer to Section 2.1. To ensure that the Mechanical Subsystem is fulfilling its responsibilities, a requirements & verification table can be found below.

Table 5: Mechanical Subsystem - Requirements & Verification

Requirements	Verification
The shell wall over each touch pad must be 2 mm thick or less.	Before sticking on the copper tape, measure the wall thickness at each of the eight pad locations with calipers. Confirm that every reading is 2.0 mm or less.
The finished pet must fit inside a 15 cm x 12 cm x 10 cm box and weigh 400 g or less.	Measure the length, width, and height of the finished pet with a ruler. Then, weigh it on a kitchen scale. Confirm that all values are within the limits.

Requirements	Verification
The pet must have no openings wider than 5 mm near the wheels or tail, and no small parts that can come off.	- Try to push a 5 mm hex key into every gap around the wheels and tail. Confirm that it does not fit. - Pull on the tail, ears, and caster by hand. Confirm that nothing comes off, or that anything that comes off is too large to fit through a 31.7 mm hole.

2.4 Hardware Design

2.4.1 Operating Voltage & Regulation

We will need to provide regulated voltage to every part of the pet from a single lithium polymer battery. Figure 6 shows the supply voltage ranges of our main parts, taken from their datasheets [3]-[10]. The microcontroller module, the motion sensor, and the real-time clock all need about 3.3 V, while the HT16K33 LED driver needs between 4.5 V and 5.5 V [6]. The tail servo is also made to run at around 5 V.

Operating conditions	Minimum	Maximum	Units
Parts on the 3.3 V rail			
ESP32-S3-WROOM-1 (VDD33)	3.0	3.6	V
LSM6DSO32 motion sensor (VDD)	1.71	3.6	V
DS3231M real-time clock (VCC)	2.3	5.5	V
Parts on the 5 V rail			
HT16K33 LED driver (VDD)	4.5	5.5	V
DRV8833 motor driver (VM)	2.7	10.8	V
Parts on the battery side			
MAX98357A amplifier (VDD)	2.5	5.5	V
TPS63021 buck-boost input (VIN)	1.8	5.5	V
TPS61023 boost start-up input (VIN)	1.8	5.5	V

Figure 6: Supply Voltage Ranges of Our Main Parts from Their Datasheets

Our battery voltage goes from 4.2 V when it is full down to about 3.0 V when it is empty. This range goes both above and below 3.3 V, so a simple linear regulator would not work for the whole discharge. Once the battery drops close to 3.3 V, a linear regulator would drop out and the microcontroller could reset. Instead, we will use the TPS63021 buck-boost converter, which keeps its output at 3.3 V for any input from 1.8 V to 5.5 V [9]. It also has a fixed 3.3 V output, so it does not need any feedback resistors.

For the 5 V rail, we will use the TPS61023 boost converter. Its output voltage is set by the resistor divider R9 and R10 using Eq. 1, where $V_{FB} = 0.595 \text{ V}$ is the feedback reference voltage from the datasheet [10].

$$V_{5V} = V_{FB} \left(1 + \frac{R_9}{R_{10}}\right) = 0.595 \text{ V} \left(1 + \frac{732 \text{ k}\Omega}{100 \text{ k}\Omega}\right) = 4.95 \text{ V} \quad (1)$$

The boost converter runs straight from the battery instead of from the charger output. The charger has a protection switch that cuts off its output for a short time if it sees around 2 A [11], and starting the motors could trip it and reset the pet. Feeding the boost converter from the battery keeps that current away from the charger. The microcontroller turns the 5 V rail on and off with its EN5 pin, and when the rail is off, the TPS61023 fully disconnects the 5 V loads from the battery. Figure 7 shows the schematic for both regulators and the on/off controller.

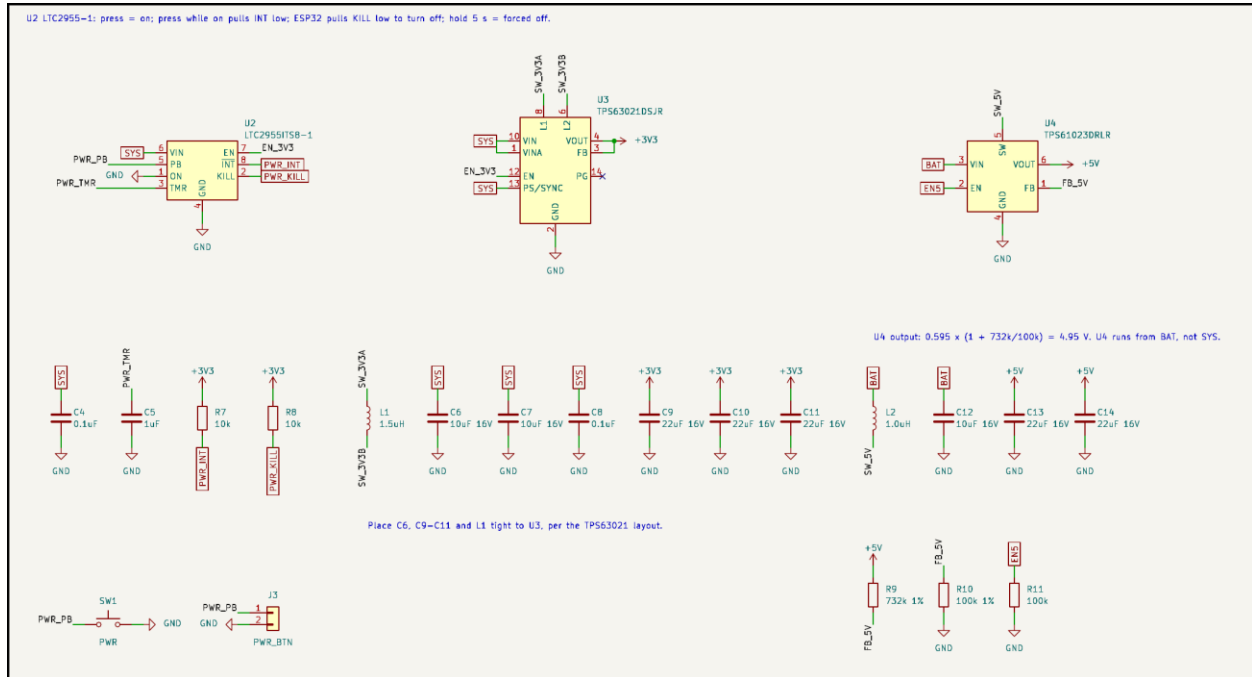


Figure 7: Schematic of the On/Off Controller, 3.3 V Buck-Boost, and 5 V Boost

2.4.2 Battery Charging

Since we are using a lithium polymer battery, we need to make sure we charge it safely, so we will be using the BQ24074 charger IC. It charges the battery from USB-C, it has a power path so the pet can run while it charges, and it has safety features mentioned in the datasheet [11] such as a battery temperature input, a safety timer, and current limiting. We set it to 500 mA USB mode so

that any laptop port or phone charger can be used. The resistor R3 on the ISET pin sets the maximum charge current using Eq. 2.

$$I_{CHG} = \frac{K_{ISET}}{R_{ISET}} = \frac{890 \text{ A}\Omega}{1.0 \text{ k}\Omega} = 0.89 \text{ A} \quad (2)$$

Since the 500 mA USB limit is lower than this, the real charge current will be about 0.45 A, and an empty 2500 mAh battery will charge in about 6 hours. A 10 k Ω thermistor next to the battery pauses charging if the battery gets too hot or too cold. A green LED shows that USB power is connected, and a red LED shows that the battery is charging. The charger schematic can be seen in Figure 8.

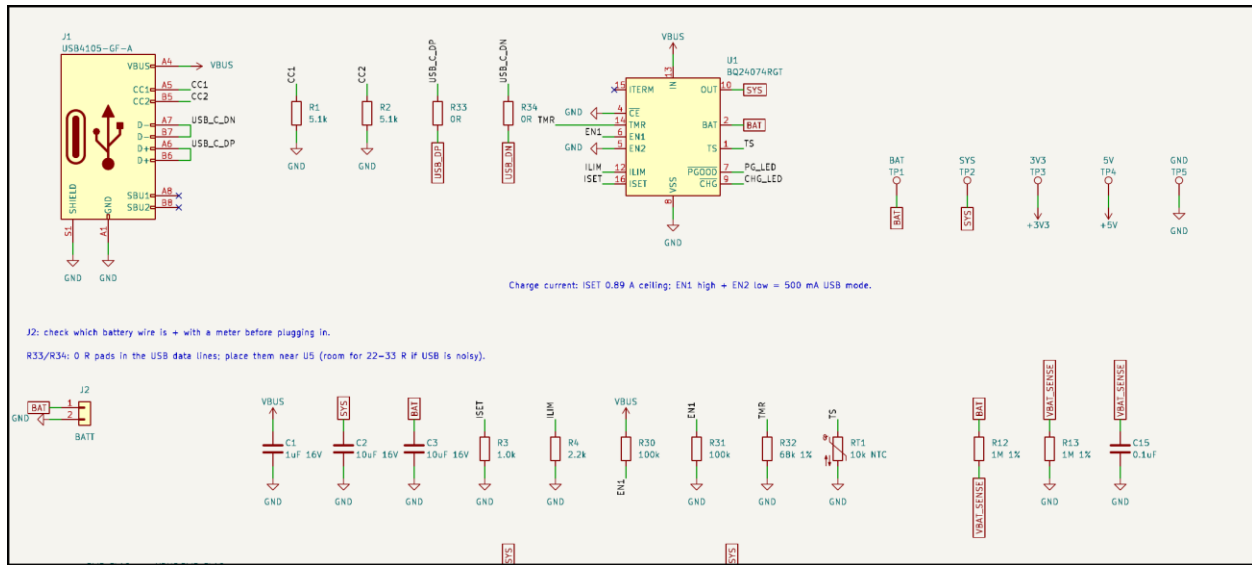


Figure 8: Schematic of the USB-C Input, BQ24074 Charger, and Battery Divider

2.5 Software Design

2.5.1 Touch Classification & Mood

The core part of our project is the software on the microcontroller that decides how the pet reacts. To do this, our microcontroller will take the following as inputs into its algorithm: the eight touch pads, the motion sensor interrupt, the power button, and the battery voltage. Our algorithm will then take these inputs and choose the right state for the pet. Figure 9 shows a flowchart of our algorithm.

The possible states are as follows:

- IDLE: No one is touching the pet. The pet shows its current mood on its face, and its mood and energy slowly drift over time. If it has been ignored for a long time, it becomes quiet and withdrawn.
- TAP: One pad was touched briefly (30 ms to 300 ms) with no neighboring pads touched. The pet blinks, chirps, and turns toward the touch.
- PET: Two or more back pads were touched in order, from front to back, within 600 ms. This is the strongest positive interaction. The pet's mood and attachment go up, its eyes squint happily, and its tail wags.
- HOLD: Two or more pads were touched at the same time for more than 1 second. The pet calms down and purrs with a low tone.
- PICKED UP: The motion sensor detected a lift or a shake. The pet immediately stops its motors, then reacts with a surprised face and chirp.
- SAVE: The power button was pressed or the battery is low. The pet turns off the 5 V rail, saves its state to flash, and turns itself off.

Every interaction moves the mood, energy, and attachment by a small amount, and each value is kept between 0 and 100. While the pet is off, each value drifts toward a resting value. When the pet turns on, it reads the time from the real-time clock and uses Eq. 3 to work out how much each value changed while it was off, where x_0 is the saved value, t is how long the pet was off, and τ is a time constant that we choose for each value.

$$x(t) = x_{rest} + (x_0 - x_{rest})e^{-t/\tau} \quad (3)$$

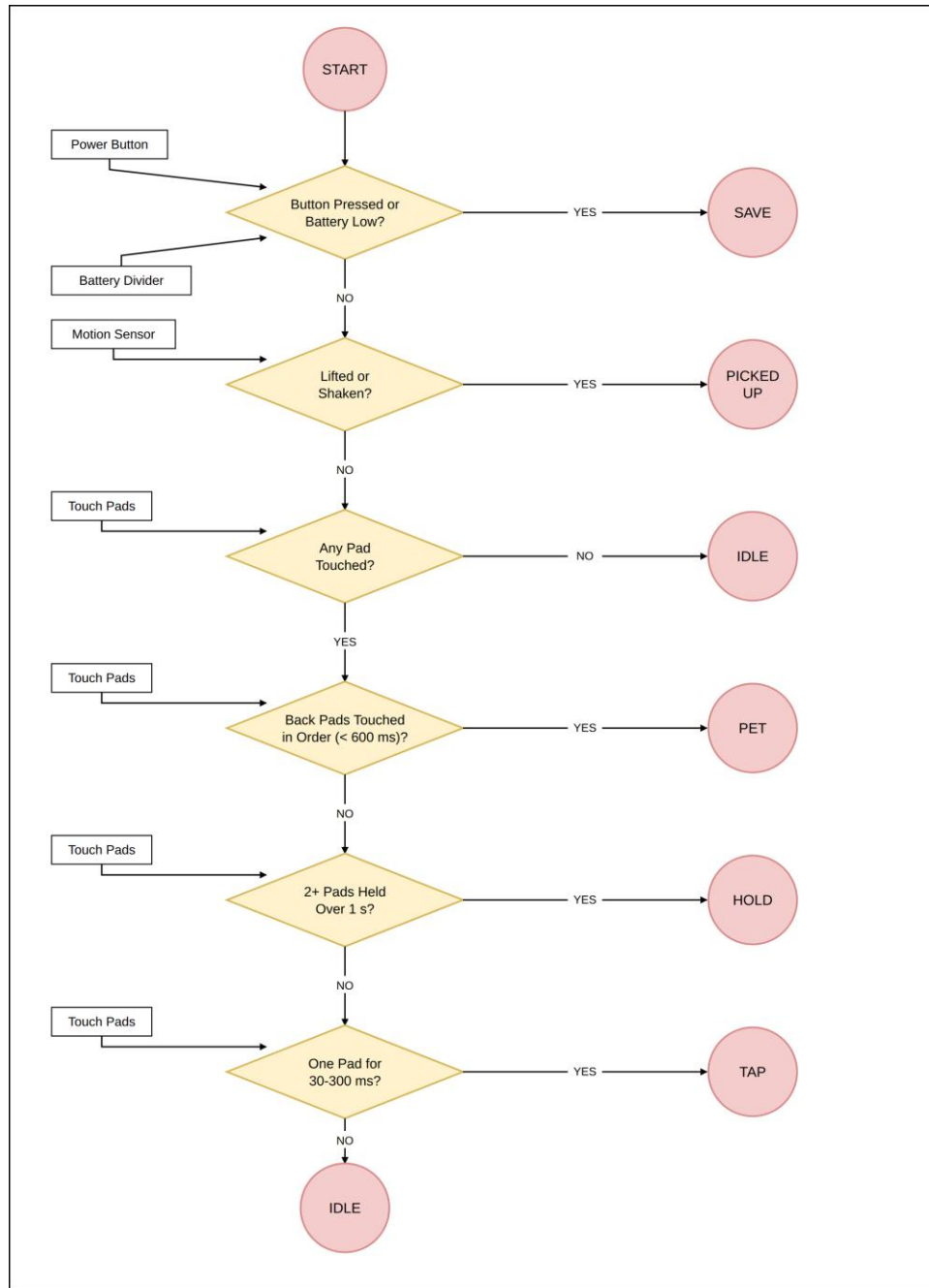


Figure 9: Touch Classification Flowchart

2.5.2 Touch Sensing

The ESP32-S3 has built-in capacitive touch sensing on 14 of its pins [3], and we will use 8 of them. For each pin, the microcontroller measures how long it takes to charge and discharge the pad, which gets longer when a finger adds capacitance. Our firmware will read all eight pads every 10 ms and compare each reading to a slowly updated baseline. If a reading moves past a set

threshold, the pad counts as touched. Updating the baseline slowly means that changes in temperature or humidity will not be seen as touches.

Figure 10 shows how a pet (a stroke along the back) looks to our firmware. The three back pads turn on one after another, from the front pad to the rear pad, about 100 ms apart. Since we read the pads every 10 ms, we can easily tell which pad turned on first, so a stroke is not mistaken for separate taps.

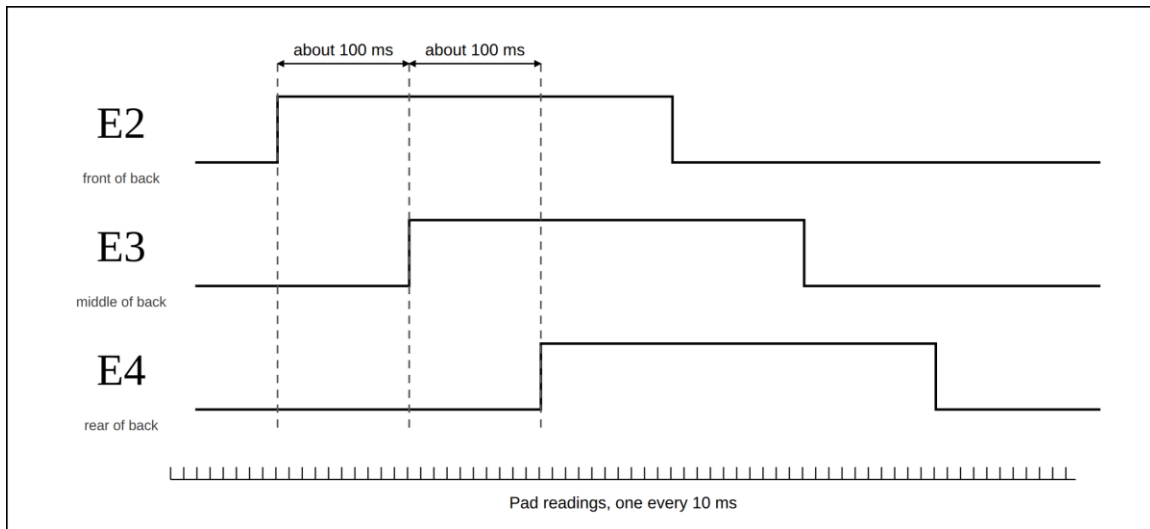


Figure 10: Touch Pad Signals for a Pet (Stroke Along the Back)

Figure 11 shows the signals for a tap and for a hold. A tap turns on only one pad for a short time, while a hold turns on two or more pads at the same time and keeps them on for longer than 1 second.

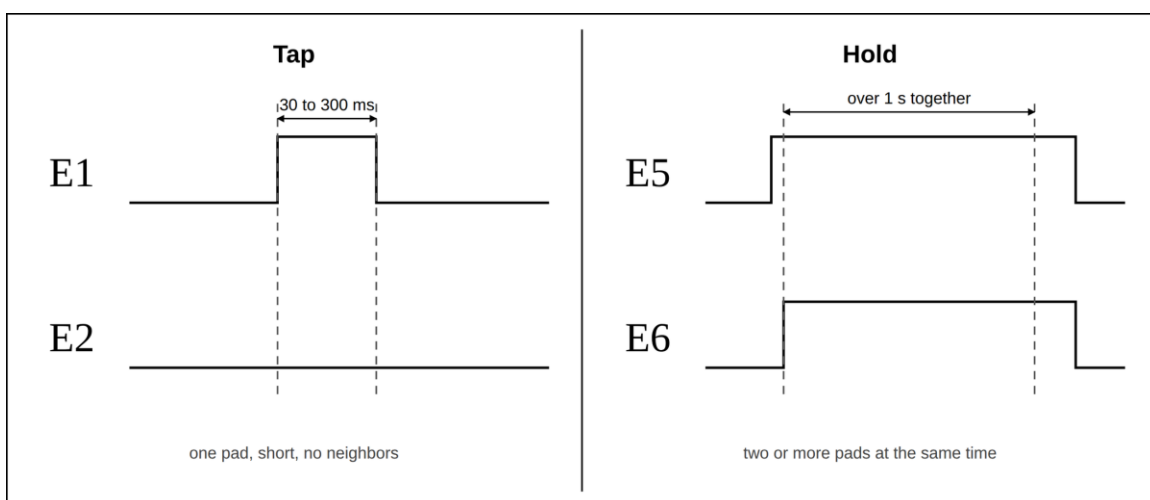


Figure 11: Touch Pad Signals for a Tap and a Hold

2.5.3 Saving the Pet's State

To let the pet remember how it was treated, we will save a small block of data into the microcontroller's flash memory. The breakdown of this block can be seen in Figure 12. Bytes (0:2) hold the mood, energy, and attachment values, each from 0 to 100. Byte (3) is reserved for future additions. Bytes (7:4) hold the time from the real-time clock when the block was saved. Bytes (9:8) hold a CRC-16 check value of the first eight bytes.

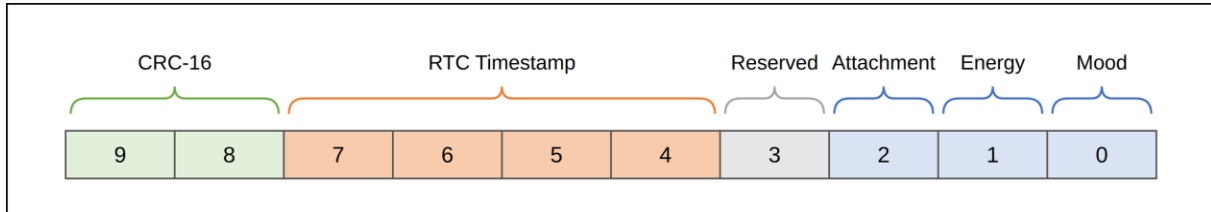


Figure 12: Byte Breakdown of the Saved State Block

There is a possibility that the pet loses power in the middle of a save, which could leave the saved data corrupted. To catch this, when the pet turns on it recomputes the CRC and compares it with the saved one. If they do not match, it loads the last good copy instead. We will use the ESP-IDF NVS library to store the block, which keeps the old copy until the new one is fully written. The pet saves whenever the power button is pressed, when the battery is low, after big changes in mood, and otherwise at most once every 10 minutes so that we do not wear out the flash.

2.6 Commercial Component Selection

2.6.1 Drive Motors

For the drive motors, we will be using a pair of N20 gearmotors with built-in encoders from SparkFun [12]. We chose these because they are small enough to fit inside a hand-sized shell, they already come with cables that plug into a JST connector, and their encoders give 882 counts per turn of the wheel, which lets us match the speed of both wheels so the pet drives straight. At 5 V the motors spin at about 417 RPM with no load, which with our 32 mm wheels gives a top speed of about 0.7 m/s. We only need 0.05 to 0.2 m/s, so the motors will never be pushed hard.

2.6.2 LED Matrix Face

For the face, we will be using Adafruit's 16x8 1.2 inch LED matrix with an HT16K33 backpack [13]. The driver chip on the backpack handles all of the LED multiplexing, so our PCB only needs

four wires to the head (power, ground, and the two I2C lines) instead of 24. The backpack already includes pull-up resistors on its I2C lines.

2.6.3 Battery

For our battery, we chose Adafruit's 3.7 V 2500 mAh lithium polymer battery [14]. It has a built-in protection circuit that cuts off the battery if it is over-charged, discharged below about 2.8 V, or shorted. It measures 60 mm by 50 mm by 7.3 mm, which fits flat under the PCB, and it weighs 50 g. Since our firmware turns the pet off at 3.6 V, we can only use about 80% of it, or about 2000 mAh. With our estimated average current of 0.25 A, this gives about 8 hours of use on one charge.

2.7 Tolerance Analysis

2.7.1 Battery Voltage Under Load

The biggest risk in our design is the battery voltage dropping too low when the LED face, the speaker, and the motors all draw current at once. If the battery drops below its 2.8 V protection cutoff before the pet finishes saving, the pet's state could be lost. The worst case is right when the low-battery save is about to happen, while the pet is driving with its face on and playing a sound. In this case, the 5 V rail supplies the LED face (0.15 A with our brightness limit) and both motors starting up (0.375 A each, scaled from the motor's 0.45 A stall current at 6 V [12]), for 0.9 A in total. Assuming the boost converter is 90% efficient, the power it draws from the battery is:

$$P_{5V} = \frac{5V \times 0.9A}{0.9} = 5.0W \quad (4)$$

Adding the speaker at 1 W (85% efficient) and the 3.3 V logic at about 0.11 A (90% efficient), the total power drawn from the battery is:

$$P = P_{5V} + \frac{1W}{0.85} + \frac{3.3V \times 0.11A}{0.9} = 6.58W \quad (5)$$

Our firmware starts the low-battery save when it reads 3.60 V. If the battery reading is off by 4%, the real battery voltage could be as low as:

$$V_{cell} = \frac{3.60V}{1.04} = 3.46V \quad (6)$$

The battery, its protection circuit, and the wires have some resistance, which we estimate as $R = 0.25 \Omega$. Since the converters draw a fixed amount of power, the voltage at the battery terminals V_T must satisfy $V_T = V_{cell} - (P/V_T)R$. Solving this for V_T gives:

$$V_T = \frac{V_{cell} + \sqrt{V_{cell}^2 - 4PR}}{2} = \frac{3.46 + \sqrt{3.46^2 - 4(6.58)(0.25)}}{2} = 2.89 \text{ V} \quad (7)$$

This is 0.09 V above the 2.8 V cutoff, so the pet can still finish saving. If the tail servo (about 0.3 A) were also allowed to move at the same time, the total power would rise to 8.25 W and V_T would drop to 2.69 V, which is below the cutoff. Because of this, our firmware will never move the tail while the motors are running. This result is also conservative, since in normal driving the motors run at less than 30% duty cycle and draw much less than their full start-up current. Once the board is built, we will measure the real battery resistance, and if it is higher than 0.25 Ω , we will raise the save threshold.

2.8 Cost Analysis

The total cost for parts as seen below in Figure 13 before shipping is \$138.22. 5% shipping cost adds another \$6.91 and 10% sales tax adds another \$13.82. Based on the average starting salary of \$119,064 for Illinois computer engineering graduates [15], or about \$57.24/hr, we can expect a salary of \$57.24/hr x 2.5 x 60 hr = \$8,586 per team member. We need to multiply this amount by the number of team members, \$8,586 x 3 = \$25,758 in labor cost. This comes out to be a total cost of \$25,916.95.

Description	Manufacturer	Part Number	Quantity	Extended Price	Link
USB-C receptacle	GCT	USB4105-GF-A	1	\$0.80	Link
Li-ion charger with power path	Texas Instruments	BQ24074RGTR	1	\$2.43	Link
Pushbutton on/off controller	Analog Devices	LTC2955ITS8-1#TRMPBF	1	\$8.61	Link
3.3 V buck-boost converter	Texas Instruments	TPS63021DSJR	1	\$3.34	Link
5 V boost converter	Texas Instruments	TPS61023DRLR	1	\$1.22	Link
Inductor 1.5 μ H	Würth Elektronik	74438357015	1	\$1.69	Link
Inductor 1.0 μ H	Würth Elektronik	74438357010	1	\$1.69	Link
Microcontroller module	Espressif	ESP32-S3-WROOM-1-N16R8	1	\$6.76	Link
Real-time clock	Analog Devices	DS3231MZ+TRL	1	\$14.68	Link
6-axis motion sensor	STMicroelectronics	LSM6DSO32TR	1	\$4.82	Link
I2C level shifter	Texas Instruments	PCA9306DCUR	1	\$0.91	Link
I2S class-D amplifier	Analog Devices	MAX98357AETE+T	1	\$4.08	Link
Dual H-bridge motor driver	Texas Instruments	DRV8833PWPR	1	\$3.02	Link
Cap 10 μ F 16 V X5R 0805	Samsung	CL21A106KOQNNNE	7	\$0.70	Link
Cap 22 μ F 16 V X5R 1206	Murata	GRM31CR61C226KE15L	6	\$2.10	Link
Cap 2.2 μ F 16 V X5R 0805	Samsung	CL21A225KOQNNNE	1	\$0.10	Link
Cap 0.01 μ F 50 V X7R 0805	Samsung	CL21B103KBANNNC	1	\$0.10	Link
Cap 100 pF 50 V C0G 0805	Samsung	CL21C101JBANNNC	1	\$0.10	Link
Thermistor NTC 10 k Ω	Murata	NCP18XH103F03RB	1	\$0.10	Link
Res 68 k Ω 1% 0805	Yageo	RC0805FR-0768KL	1	\$0.10	Link
Res 732 k Ω 1% 0805	Yageo	RC0805FR-07732KL	1	\$0.10	Link
Res 200 k Ω 1% 0805	Yageo	RC0805FR-07200KL	1	\$0.10	Link
Conn header 2-pos 2 mm (battery, speaker)	JST	B2B-PH-K-S(LF)(SN)	2	\$0.34	Link
Conn header 6-pos 1 mm (motors)	JST	BM06B-SRSS-TB(LF)(SN)	2	\$1.00	Link
Pin header 1x40 2.54 mm	Sullins	PRPC040SAAN-RC	1	\$1.23	Link
16x8 LED matrix + HT16K33 backpack	Adafruit	2037	1	\$14.95	Link
Battery LiPo 3.7 V 2500 mAh	Adafruit	328	1	\$14.95	Link
N20 motors with encoders (pair)	SparkFun	ROB-28633	1	\$19.95	Link
Wheels 32 x 7 mm (pair)	Pololu	1087	1	\$4.45	Link
Ball caster 3/8 inch	Pololu	951	1	\$2.95	Link
SG92R micro servo	Adafruit	169	1	\$5.95	Link
Speaker 4 Ω 3 W	Adafruit	3351	1	\$3.95	Link
Copper foil tape 6 mm	Adafruit	1128	1	\$5.95	Link
PLA filament for shell (about 200 g)	Any	n/a	1	\$5.00	
Resistors, 0.1 and 1 μ F caps, LEDs, buttons, test points	ECE self-service drawers	Various	60	\$0.00	
2-layer PCB	PCBWay (course order)	n/a	1	\$0.00	
Total (before shipping and tax)				\$138.22	

Figure 13: Itemized list of Components and Costs

2.9 Schedule

Week	Task	Person
October 5th - October 11th	Design review	Everyone
	Finish PCB layout & pass audit	Megna
	PCB ORDER OCTOBER 8TH	Everyone
	Order parts from DigiKey and Adafruit	Neel
	Start shell and drive chassis CAD	Vinny
October 12th - October 18th	Breadboard demo	Everyone
	Write power test firmware (EN5, KILL, battery reading)	Megna
	Touch sensing on an ESP32-S3 dev board	Neel
	Print a 2 mm test wall and test touch through it	Vinny
October 19th - October 25th	Stencil and reflow the first PCB	Megna
	Mood model, flash saving, and real-time clock code	Neel
	Print first full shell and drive chassis	Vinny
October 26th - November 1st	Power bring-up and power tests, revisions to PCB design	Megna
	PCB ORDER OCTOBER 29TH (if needed)	Everyone
	Move firmware to our PCB: I2C, motion sensor, face, audio	Neel
	Motor driving with encoder speed matching, tail linkage	Vinny
November 2nd - November 8th	Progress demo	Everyone
	Control and sensing tests on the PCB	Megna
	Pick-up detection and low-battery save	Neel
	Mount PCB, battery, face, and speaker in shell; add touch pads	Vinny
November 9th - November 15th	Battery voltage and battery life tests	Megna
	Happy, neutral, and lonely expressions	Neel
	Mechanical tests: wall, size, weight, pinch points	Vinny
November 16th - November 22nd	Mock demo & mock presentation	Everyone
	High level requirement tests, fix existing minor bugs	Everyone
November 23rd - November 29th	Fall break	Everyone
November 30th - December 6th	Final demo & final presentation	Everyone
December 7th - December 10th	Final paper & lab checkout	Everyone

Figure 14: Schedule for Project Progression

2.10 Risk Analysis

Purrceptron is meant to be held and handled, so the parts that could hurt someone or damage the pet are the most important to get right. When focusing on the components we are building ourselves, the most important is the power system, since it is connected to a lithium polymer

battery. If the charger or regulators fail, the battery could be over-charged or shorted. To reduce this risk, we use a battery with its own protection circuit, our charger has a temperature input and a safety timer, and we will never charge the battery unattended. The next most important part is the drive system. If the motion sensor fails to detect that the pet was picked up, the wheels could keep spinning in someone's hands. To reduce this risk, the pet drives slowly (0.2 m/s at most), each drive is limited to 50 cm, and the motors are too small to hurt a finger. Finally, should the firmware freeze, holding the power button for about 5 seconds will force the pet off, since the LTC2955 cuts power on its own after a long press [16]. In short, we will try to make sure that any single failure leaves the pet in a safe state.

3 Ethics and Safety

Our project aims to give people who cannot own a pet some of the comfort of having one, at a low cost. With this project we are putting the safety and welfare of the user first, which correlates to IEEE's Code of Ethics Section I.1, which is "to hold paramount the safety, health, and welfare of the public" [17]. The same section also asks us "to protect the privacy of others." Purrceptron has no camera, microphone, or network connection, and we will turn off the ESP32-S3's Wi-Fi and Bluetooth, so nothing a user does with the pet ever leaves the device.

We will also not claim that Purrceptron is a medical or therapy device. It is a companion toy, and we will be "honest and realistic in stating claims or estimates based on available data" as in IEEE's Code of Ethics Section I.5 [17]. Since the pet reacts to being ignored, we were also careful that it should not pressure people. When it is neglected, it only becomes quiet and withdrawn, and it never acts in a way meant to make the user feel guilty.

Additionally, there are concerns with the use of lithium polymer batteries. These batteries can be a fire hazard, especially if they are over-charged, shorted, or damaged. To reduce this risk, we use a battery with a built-in protection circuit [14], our charger limits the charge current and stops charging if the battery is too hot or too cold, and we will follow the course battery safety guidelines [18]. We will never charge the battery unattended, we will charge it inside a fire-safe bag, and we will stop using any battery that swells or gets hot. We also follow the standards that apply to our design: the USB Type-C specification for how the pet draws power from a USB port [19], the IEC 62133-2 safety standard for lithium cells, which is why we buy a protected cell from a known supplier instead of building our own pack [20], and the small parts test in 16 CFR Part 1501, which we use to make sure no part that comes off is a choking hazard [21].

By undertaking this project, we hope to show that a low-cost robot companion can respond to touch in a meaningful way. It works offline anywhere in the world, it does not depend on any language since it only uses lights, sounds, and movement, and it uses one rechargeable battery instead of disposable ones. Its shell screws together and every part is standard, so it can be repaired instead of thrown away.

References

- [1] Office of the U.S. Surgeon General, "Our epidemic of loneliness and isolation: The U.S. Surgeon General's advisory on the healing effects of social connection and community," U.S. Department of Health and Human Services, 2023. [Online]. Available: <https://www.hhs.gov/sites/default/files/surgeon-general-social-connection-advisory.pdf>
- [2] W. Moyle, C. J. Jones, J. E. Murfield, L. Thalib, E. R. A. Beattie, D. K. H. Shum, S. T. O'Dwyer, M. C. Mervin, and B. M. Draper, "Use of a robotic seal as a therapeutic tool to improve dementia symptoms: A cluster-randomized controlled trial," *Journal of the American Medical Directors Association*, vol. 18, no. 9, pp. 766-773, 2017. [Online]. Available: [https://www.jamda.com/article/S1525-8610\(17\)30189-5/pdf](https://www.jamda.com/article/S1525-8610(17)30189-5/pdf)
- [3] *ESP32-S3-WROOM-1 & ESP32-S3-WROOM-1U Datasheet*, Espressif Systems. [Online]. Available: https://documentation.espressif.com/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf
- [4] *LSM6DSO32 iNEMO Inertial Module*, STMicroelectronics. [Online]. Available: <https://www.st.com/resource/en/datasheet/lsm6dso32.pdf>
- [5] *DS3231M ± 5 ppm, I2C Real-Time Clock*, Analog Devices. [Online]. Available: <https://www.analog.com/media/en/technical-documentation/data-sheets/DS3231M.pdf>
- [6] *HT16K33 RAM Mapping 16x8 LED Controller Driver with Keyscan*, Holtek Semiconductor, 2011, rev. 1.10. [Online]. Available: <https://cdn-shop.adafruit.com/datasheets/ht16K33v110.pdf>
- [7] *DRV8833 Dual H-Bridge Motor Driver*, Texas Instruments. [Online]. Available: <https://www.ti.com/lit/ds/symlink/drv8833.pdf>
- [8] *MAX98357A/MAX98357B Tiny, Low-Cost, PCM Class D Amplifier with Class AB Performance*, Analog Devices. [Online]. Available: <https://www.analog.com/media/en/technical-documentation/data-sheets/MAX98357A-MAX98357B.pdf>

- [9] *TPS6302x High Efficiency Single Inductor Buck-Boost Converter*, Texas Instruments. [Online]. Available: <https://www.ti.com/lit/ds/symlink/tps63021.pdf>
- [10] *TPS61023 3.7-A Boost Converter with 0.5-V Ultra-Low Input Voltage*, Texas Instruments. [Online]. Available: <https://www.ti.com/lit/ds/symlink/tps61023.pdf>
- [11] *BQ2407x 1.5-A USB-Friendly Li-Ion Battery Charger and Power-Path Management IC*, Texas Instruments. [Online]. Available: <https://www.ti.com/lit/ds/symlink/bq24074.pdf>
- [12] *N20 Motor with Encoder and Cable (Pair)*, ROB-28633, SparkFun Electronics. [Online]. Available: https://cdn.sparkfun.com/assets/a/3/9/9/9/28633_datasheet_N20_Motor_With_Encoder_and_Cable_pair.pdf
- [13] *Adafruit LED Backpacks: 1.2" 16x8 Matrix*, Adafruit Industries. [Online]. Available: <https://learn.adafruit.com/adafruit-led-backpack/1-2-16x8-matrix-arduino-setup>
- [14] *Lithium Ion Polymer Battery - 3.7V 2500mAh*, Adafruit Industries, product 328. [Online]. Available: <https://www.adafruit.com/product/328>
- [15] *Salary Averages*, Department of Electrical and Computer Engineering, University of Illinois Urbana-Champaign. [Online]. Available: <https://ece.illinois.edu/admissions/why-ece/salary-averages>
- [16] *LTC2955 Pushbutton On/Off Controller with Automatic Turn-On*, Analog Devices. [Online]. Available: <https://www.analog.com/media/en/technical-documentation/data-sheets/2955fa.pdf>
- [17] *IEEE Code of Ethics*, IEEE, 2020. [Online]. Available: <https://www.ieee.org/about/corporate/governance/p7-8.html>
- [18] *ECE 445 Safety Guidelines*, University of Illinois Urbana-Champaign. [Online]. Available: <https://courses.grainger.illinois.edu/ece445/guidelines/safety.asp>
- [19] *Universal Serial Bus Type-C Cable and Connector Specification*, USB Implementers Forum. [Online]. Available: <https://www.usb.org/document-library/usb-type-cr-cable-and-connector-specification-release-24>

- [20] *IEC 62133-2:2017, Secondary Cells and Batteries Containing Alkaline or Other Non-Acid Electrolytes - Safety Requirements for Portable Sealed Secondary Lithium Cells, and for Batteries Made from Them, for Use in Portable Applications - Part 2: Lithium Systems*, International Electrotechnical Commission, 2017.
- [21] *16 CFR Part 1501, Method for Identifying Toys and Other Articles Intended for Use by Children Under 3 Years of Age Which Present Choking, Aspiration, or Ingestion Hazards Because of Small Parts*, U.S. Consumer Product Safety Commission. [Online]. Available: <https://www.ecfr.gov/current/title-16/chapter-II/subchapter-C/part-1501>