**Background**

Icarus ONE is truly the culmination of all the learning and work I’ve done with microcontrollers, circuit design, and programming over the last 4 years. A year and a half ago, I started my first high-altitude balloon project. I basically completed all that I had intended to accomplish, both hardware and software, but I wasn’t able to launch with the move of my Ph.D. laboratory from Houston, TX to Bloomington, IN at that time. Now, I couldn’t be happier that my first prototype now sits in a storage unit, as this second attempt revealed too many coding blunders and potential hardware problems to count, but hey…I’m self-taught, and that’s part of learning!

**Rationale**

Now, I’m far more prepared to take on this project – probably far too ambitious as well. I don’t in any way recommend taking the “go big or go home” approach I did in this project, but I do hope that people can learn from my mistakes in one or more aspects of their own projects that I’ve included here.

So, as my opus magnum, Icarus ONE includes as many elements of my knowledge as I could integrate (and/or justify including). I’ll try to lay it all out simply and concisely, so others may hop onto the learning curve I was able to navigate.

**Overview**

Any high-altitude balloon has at least two critical elements, GPS location and some form of communicating this to someone tracking it as it falls back to earth with (hopefully) some really awesome media captured in near-space! Icarus contains a UBLOX GPS from this purpose, and I chose to opt for redundancy with communication, including both a 70cm band transmitter for radio teletype (RTTY) and a GPRS shield for SMS messaging. By including SMS capabilities, I was also able to add features like the piezo buzzer for locating Icarus in high-grass or similar situations, which can be activated by sending a text to the device after landing.

Outside of the core functions, all of my other add-ons were optional, but I’m excited about looking at the data afterwards, maybe even gaining a better understanding of the atmosphere (I’m also a weather nerd). I’ve added external environmental sensors, such as temperature, humidity, gas, and atmospheric pressure as well as internal sensors such as the Adafruit 1604 11-DOF (barometer, gyroscope, accelerometer, and magnetometer) and a sensor for battery temperature monitoring.

Some of the devices also provide output. For example, a custom heater board provides a more stable environment for the lithium ion batteries used.

The real payoff for a high-altitude balloon project is with the pictures. I mean, why send something up to around 100,000ft if you can’t see what it looks like up there!? [I’ll post later about the custom rockets I’ve also been working on in parallel with the iterations of Icarus ONE.] Icarus is equipped with three cameras: an up-facing webcam for capturing the balloon and parachute during certain flight phases, a down-facing webcam for capturing the earth from straight overheard, and the PiCam V2 which faces outwards and hopefully will capture the curvature of the earth from the edge of space. As an added bonus, a servo motor has been setup to swing a “selfie” of my girlfriend and me in front of the PiCam near the peak altitude…I’ve always wanted a picture of us in space!

Finally, there were a few personal items sent up, both from me and from a couple others to whom I offered the opportunity to send something. From me, I chose something very personal. Sadly, my father passed away a year and a half ago. He was a flight attendant for forty years and used to fly remote control gliders back in the early eighties, so much of his life was spent in the sky. So, it seemed like a perfect fit to include some of his ashes as well, which will be released as high as I’m able to get them, where our world meets the infinite expanse of space.

So, here we go. Time to get down to the nitty-gritty of Icarus’ design, construction, and programming.

**Hardware (Enclosures & Power)**

**Payload boxes:**

Luckily, biological laboratories have more Styrofoam boxes than anyone knows what to do with, so I had my choice of all shapes, sizes, and wall thicknesses. While Icarus ONE started with a single payload container (the larger box sitting on top), it quickly became necessary to expand this, as all of the things I crammed into this project required a decent amount of power. To provide this, I needed to expand Icarus with a smaller Styrofoam box underneath the main one to hold the batteries necessary for operation. The two boxes are permanently attached to each other, and holes were ported between the two to allow power cords, sensor wires, and antenna cables to pass through.

**Lower Container (Power & Environmental Control):**

All of the power sources can be found in the lower container.

1. Main, low-current power for the Raspberry Pi 3, Arduino Mega, and Arduino Uno is provided by two 10,000mAh portable charging packs (Anker brand from Amazon), which have USB ports for convenient powering of the devices.
2. The Arduino Nano (which drives a servo motor) and a few accessories are powered by two sets of 4x NCR18650B lithium ion batteries each in series and linked together with a custom parallel harness made out of parts I used to recharge my drone batteries. [Fun drone posts to come later!] This arrangement creates a “4S2P” lithium ion setup, with a nominal 14.8V and 6800mAh (rated) of power. Each of the lithium ion battery packs was wired with a regulator board that prevents over-charging, excessive current draw, and excessive battery drain.

Looking at the data sheet for 18650 batteries, the minimum temperature under discharge conditions was -20C. The payloads are well insulated, but it’s definitely possible for temperatures to drop below this at high altitudes, potentially even to -40C. My calculations showed more than sufficient battery capacity to handle all of the main functions with significant overhead, so I chose to include a heater from the battery payload bay to keep the lithium ions happy for the journey. To do this, I simply wired up two 10ohm (10W) resistors in series on a custom board mounted in the lower payload container. Also in the container is a DS18B20 temperature sensor, which monitors the battery box and relays the temperature back to the Arduino Mega via the one-wire protocol. When temperatures below 0C are detected, a relay is tripped, allowing 14.8V power from the lithium ions to pass through the resistors, heating them up nicely.

Next up, the guts and brains of Icarus ONE…

**Arduino Mega (The Sensor & Communication Brain):**

For the main data acquisition board, I chose to use an Arduino Mega. I’m sure there was a way to pull all of this off without using the Mega, but it’s just so convenient with four serial ports that eliminate the need for software serial and the multitude of analog inputs that were really useful given the six gas sensors used with two analog pins already used by the I2C devices.

(Question #1: Can I2C analog pins be used as analog input and I2C simultaneously? Never investigated, but I suspected not. Question #2: Is there any SPI, I2C, or other alternate protocol device that can multiplex analog inputs? For example, is there some sort of board that can accept many analog inputs and send them over a couple wires to a device [with limited analog inputs like the Uno] using an alternate protocol?)

The Mega serves at the main input for all of the sensors, internal and external, output and storage of data to an SD card, controls digital output to other devices, and controls media acquisition from the Raspberry Pi over a serial connection.

The SD shield I used was designed for the Arduino Uno, which has a different set of SPI pins than the Arduino Mega. To make everything work, I stacked the SD shield and the relay shield together, separate from the Mega. I then routed pins 4, 5, 6, and 7 (to trigger the four independent relays) to digital pins on the Mega, pin 9 to another digital pin to allow software power-up of the GPRS shield, and I routed the SPI pins to their proper targets as listed on the table below.

|  |  |  |
| --- | --- | --- |
| **SD Shield (Uno Compatible)** | **SPI Pin** | **Arduino Mega** |
| 10 | SS | 53 |
| 11 | MOSI | 51 |
| 12 | MISO | 50 |
| 13 | SCK | 52 |

In addition, I linked the serial jumper pin-outs on the GPRS directly to the Mega via one of its hardware serials.

Internal:

A few sensors are included internally within the main payload package. First, there’s the Adafruit 1604 11-DOF sensor breakout (**INCLUDE LINK**), and it’s fantastic. There are way better applications for this one, but it had the barometer built in, so I included it. On a single board, Adafruit was able to put an accelerometer, gyroscope, and magnetometer, which provides 6-axis data logging and will allow me to model the movement of the payload box during the trip once I (hopefully) recover it after landing. In addition, the BMP180 provides temperature and barometric pressure data that is accurate and sensitive enough to calculate altitude with +/-10cm resolution.

(Question: If a Styrofoam payload is sealed securely on the exterior, not allowing free passage of air, does any/enough air pass through the pores in the Styrofoam to allow proper operation of and altitude calculation by the BMP180? I haven’t found an answer, so I went ahead and ported a few tiny holes in the side of the box using 14Ga magnet wire.)

The other internal “sensor” is the Adafruit Ultimate GPS Breakout V3, which also has a spot to install a coin cell battery on the backside for quicker GPS lock after resets. Fortunately, the GPS breakout also has a U.FL connector where I was able to connect a U.FL to SMA adaptor. I hot-glued the SMA connector to the side of the box and can connect a 28dB active GPS antenna that is fixed to the lid of the box, optimizing reception. Since I used a Mega, I chose to use another hardware serial for communication.

External:

On the lid are six “MQ”-type gas sensors, with two custom boards accommodating three each. The boards allow the sensors to quickly plug in and be swapped out, and also have a 5V voltage regulator circuit onboard, so I can power the current-hogging gas sensors that require a heating element to function properly. I’ve had some experience with these in the past, so I bought a cheap assortment of them from Amazon (**INCLUDE LINK**) and settled on what seemed like the most useful six of the bunch.

MQ-type Sensors:

* MQ-2 🡪 Methane, Butane, Liquid Propane Gas & Smoke
* MQ-4 🡪 Methane & Compressed Natural Gas
* MQ-7 🡪 Carbon Monoxide
* MQ-8 🡪 Hydrogen
* MQ-9 🡪 Carbon Monoxide & Flammable Gasses
* MQ135 🡪 Benzene, Alcohol & Smoke

Also on the lid, under a protective covering made from a pill bottle, I included a DHT22 temperature/humidity sensor. Originally, I had an SHT11…but an unfortunate and rather dumb mistake led to me frying the little guy.

On the side of the main payload box is a second barometer, the MS5607, a functionally equivalent alternative to the BMP180 used on the internal Adafruit board. To protect from light and moisture, as directed by the datasheet, I built a protective hood out of Gorilla tape and hot glue. Really, the inclusion of this component was fairly arbitrary, but it will be interesting to see how it compares with the BMP180 under various harsh conditions on the outside, rather than the comfy interior of the box.

Other than the previously mentioned external antennae for the GPRS and GPS, the final component controlled by the Mega is the piezo buzzer. Actually, it’s a “siren,” which as far as I can tell is a piezo buzzer that oscillates in frequency to create a siren effect. It was the loudest I could find at Radio Shack, so that’s what I chose. The buzzer is activated by triggering a relay on the relay shield which provides power directly from the lithium ion batteries.

**Raspberry Pi 3 (The Media Brain):**

ABCDEF

**Arduino Nano (The Servo Controller):**

ABCDEF

**Arduino Uno (The Heartbeat Monitor):**

ABCDEF

**Power Regulation & Distribution:**

* 12V Regulator
* 5V Regulator
* 5V Distribution
* Relay Junction

**Communication:**

* RTTY
* GPRS/SMS Functions

**User Input:**

* Sensor mounting
* LED indicators
* Activation switch

**Code**