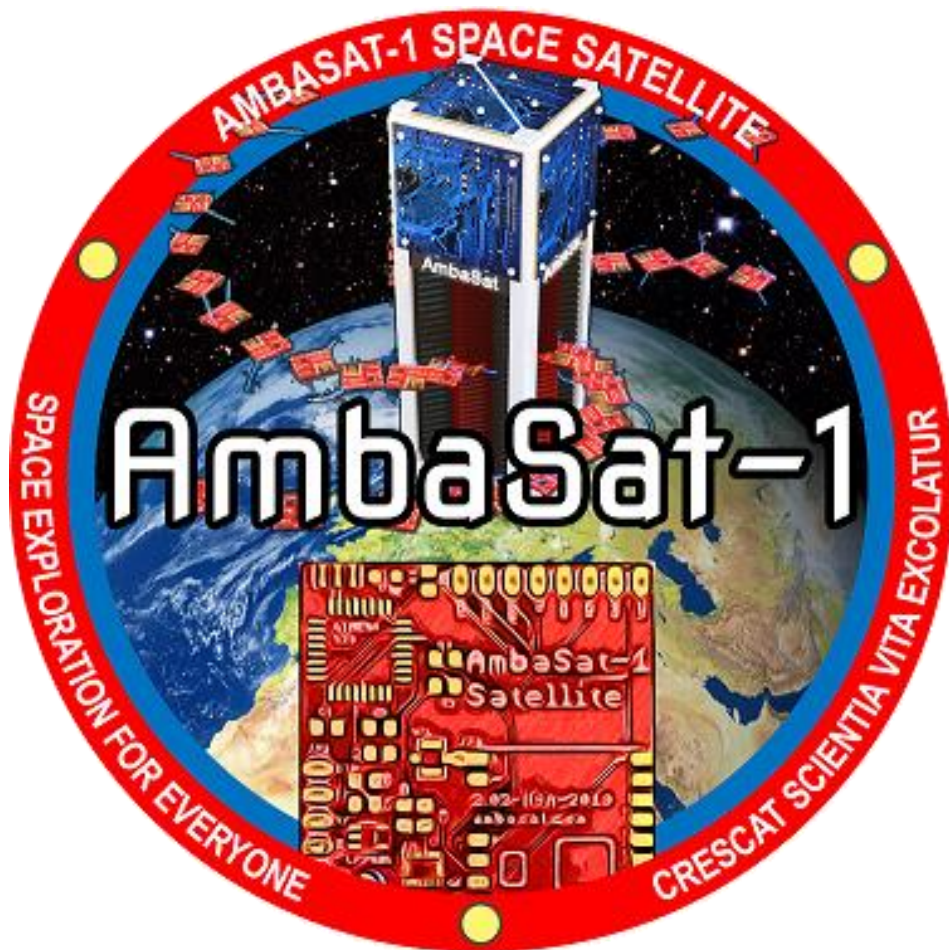


AmbaSat-1

How To Guide



Written by N Walls

December 2020

Revision number	Date	Details
1.01	06/01/2021	Initial revision
1.02	14/01/2021	Sensor numbers and formatting. 'boards' folder location.

List of contents

Introduction: What is AmbaSat-1?

What's in the Box: What is a MK-II Kit

What You'll Need: Tools, Materials & Software

Health, Safety & Precautions

Prep your Stations: Cups of tea at the ready!

Visual Studio Code: Where to get it

Step One: Slap a Tail on it

Step Two: Programmer Pins

Step Three: PCB-LED Test

Errors & Fault Finding

Step Four: Sensor Mounting

Step Five: Digital Workspace & Sensor Testing

Step Six: That's LoRaWAN!

Introduction

What is AmbaSat-1?

In this introduction we aim to provide you with a simple, watered down, run through of what the AmbaSat-1 is, how it works, and who it's for. For more details, please check out our website:

www.ambasat.com

AmbaSat-1 is easy access Space exploration. With this open-source project we will provide access to example code which is easy to edit, or you can write your own if you're savvy enough and fancy more of a challenge. We primarily use the C/C++ programming language but the AmbaSat-1 is also compatible with Arduino. You may not be a coder, but for a first-time project it's very accessible.

Aren't Satellites Big? The AmbaSat-1 is a 'Sprite' Class satellite, also known as a "Chip sat"; thanks to improvements made in semiconducting technology, we've turned the bulky technology of yesterday into the 35mm² Sprite of today.

As for what you can do with it, this is YOUR satellite, it is purpose built and coded by you! We have a range of eight sensors. Feel free to be creative and surprise us with your ideas! The basic range of available sensors are:

1. Sensor 01: Humidity and Temperature
2. Sensor 02: Temperature
3. Sensor 03: Gas & Pressure
4. Sensor 04: Ambient Light
5. Sensor 05: TVOC
6. Sensor 06: UV
7. Sensor 07: CO2 emissions
8. Sensor 08: GPS (Earthbound only)

Or you can design, build, and program your own sensor, again if you're savvy enough.

There are many practical applications for AmbaSat-1, especially for universities on a budget. The closest thing you'll find available on the market is a CubeSat but they are expensive and can take years to put together. With AmbaSat-1, you can be fully assembled and launched for the price of a PlayStation with assembly time lower than most game time experience! Also, AmbaSat-1 assembly time is less than the paying time of most games!

Do you want a satellite swarm to track climate change? With AmbaSat-1 you'll have a fraction of the foot print of the current standard of satellite launches because up to 200 AmbaSat-1 Sprite Satellites are launched into space on each rocket!

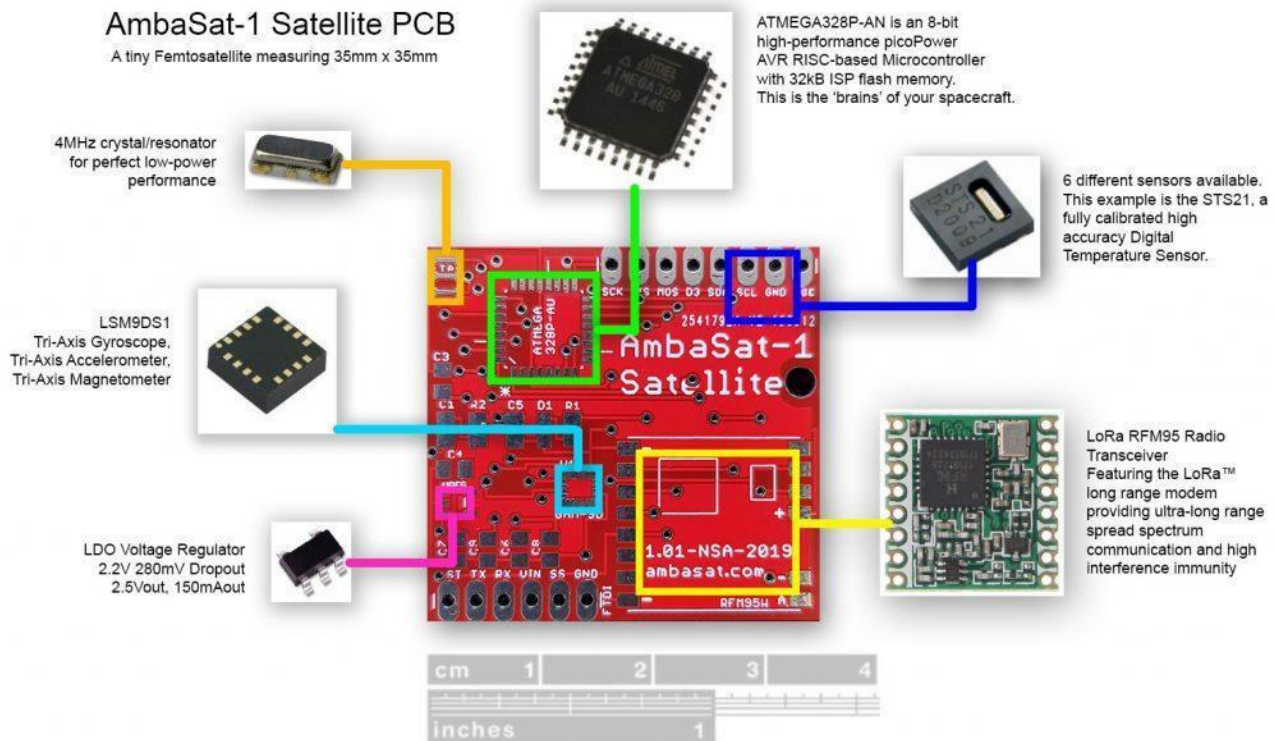
Data communications: AmbaSat-1 uses LoRaWAN and the proven transceiver range is over 700 kilometres (line of sight).

Once you've constructed and programmed your AmbaSat-1 kit, it will be sent to a commercial launch provider for packing into the payload launcher. Roughly 200 sprites can fit in the 3U launch container which will get deployed into a 310km LEO (Low Earth Orbit). Why a LEO of 310km? Deploying into the

thermosphere allows atmospheric data to be collected more reliably while also keeping mission time relatively appropriate to academic project time scales.

AmbaSat-1 is also well suited to earth bound applications and its GPS sensor is a brilliant example of this.

AmbaSat-1 Main Components



AmbaSat-1 is tiny space satellite kit that you assemble and code yourself. Once your satellite kit is assembled and programmed, it will be launched onboard a commercial rocket which will deploy your satellite into Low Earth Orbit, where it will spend up to 3 months in space.

1. **Micro Controller Unit-** this is the processor where your code will be uploaded and run from. Essentially, it's the brains of your Satellite.
2. **Tri Axis SMD-** Gyroscope, Accelerometer, Magnetometer. This helps to determine AmbaSat-1's trajectory and is also useful for earth bound tasks too such as enabling power saving measures if and when solar power is not an option.
3. **Transceiver-** The radio communications chip. This LoRaWAN chip is the impressive little component which makes use of TTN (The Things Network), which we'll cover later. This chip is capable of communicating with over 16,000 groundstation receivers here on earth. This means no specialist or expensive radio receiving equipment is needed!

What's In The Box?!

What is in the box? What comes with the MK-II kit? Let us take a look!



The Box contains:

1. Ambasat-1 PCB
2. CH340 Programmer
3. Chosen Sensor
4. Solar Board & solar panels
5. Battery holder
6. Sticker!
7. Useful reading
8. Launch Certificate
9. Bag of Components

The MK-II is less intimidating than our MK-I “Engineering Kit” and has many of its essential, fiddly and more delicate components pre soldered to the PCB, this not only reduces assembly time by a few hours, but also makes building a kit much more accessible to everyone!

Content details:

CH340 programmer: This easily connects Via USB to your own computer for coding and programming with Visual Studio or other compatible software and comes in useful for testing too. Windows users may need a CH340 ‘driver’. If so, follow the link below:

<https://learn.sparkfun.com/tutorials/how-to-install-ch340-drivers/all>

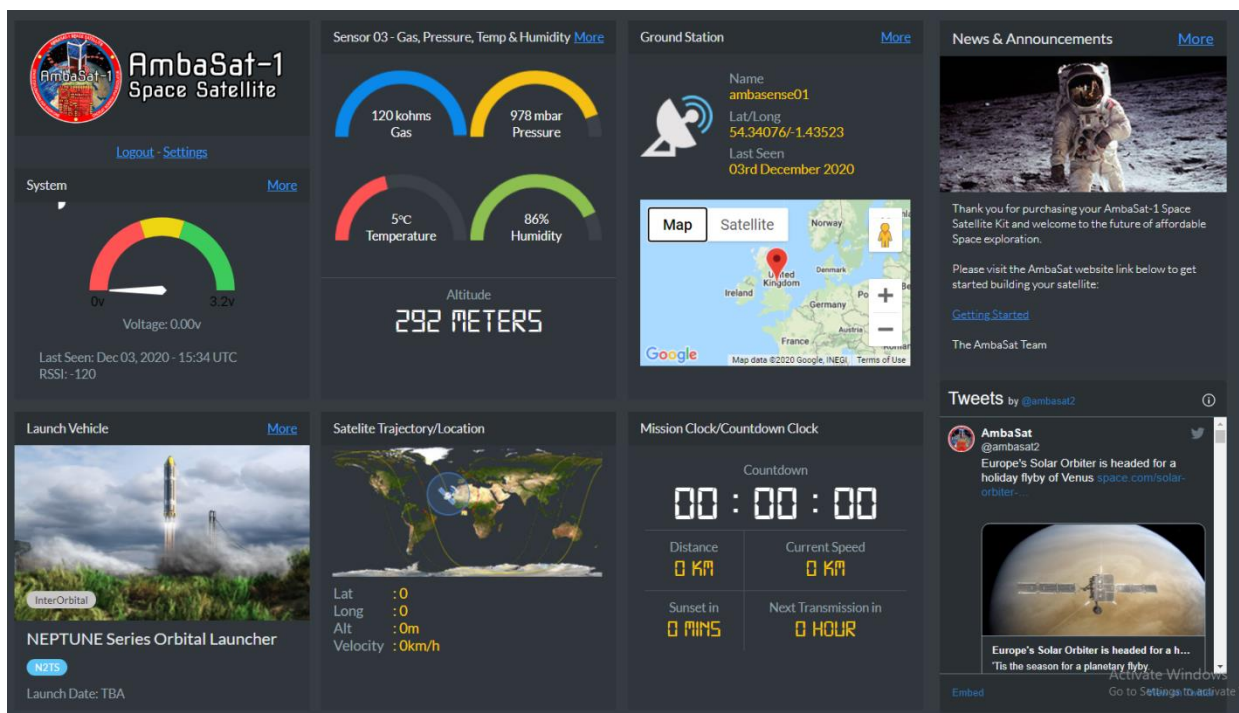
Chosen Sensor: Again, less intimidating than our Mk-I “Engineering Kit”, your chosen Sensor is supplied on a sensor ‘daughterboard’ ready for mounting onto the main AmbaSat-1 board and then it’s ready for use as you see fit!

SolarBoard: Supplied with the Mk-II, these are partially assembled solar cells (Cells in place). If you purchase a rocket kit, we recommend sending this board back to us for final checks and assembly prior to launch.

Battery holder: This allows for earth-bound testing, allowing you to test your Satellite is working fully after assembly & coding. Or this can be used to power the sensor on earth where sunlight is limited.

The sticker: ...is Sticky! We hope to see the more creative, most bizarre and best placed Ambasat-1 Stickers we provide with each kit! Keep us posted!

Reading materials: Information that will allow you to access the AmbaSat Dashboard and view data related to AmbaSat-1 and its sensors. The Dashboard can be accessed from any web browser using an internet connection.



What You Will Need!

If you already have experience in engineering and/or tinkering as a hobby or just in general, you may already have many, if not all the tools required, such as a soldering station. However, a more kitted-out work station may provide you with easier solder methods or access to different techniques that can be practiced in order to achieve the final result. Here, we will just cover the basics and list the bare essentials of what will be needed to complete assembly of the AmbaSat-1 kit.

If this is your first venture into soldering, a link will be provided to a recommended tool kit that'll help get your satellite assembled:

<https://www.amazon.co.uk/Soldering-Multimeter-NO-Soldering-Welding-Cutter/dp/B07F2XNTWV/>



Essential Tools:

- **Soldering Iron or Station** - A soldering iron is going to be your main and most reliable tool for mounting and connecting SMD (Surface Mounted Devices) and other components to your PCB. Soldering stations give much more control over the temperature of your iron, as well as often having a SMD Rework Airflow system built in.
- **Solder** - There's no getting away from needing a decent solder, for this guide I'll be using leaded solder.
- **Tweezers** (Metal or Ceramic) - Often handy for feeding solder into the right place, holding components in position or to give something a nudge in the right direction. Avoid Plastics.
- **Snips or cutting pliers** - useful for cutting your connection pins/material and trimming any wires used in the assembly.
- **A Desktop or Laptop Computer**- Needed for programming, testing and to receive satellite data from the AmbaSat Dashboard. Windows, Linux or Mac.

Recommended Software

- **Visual Studio**- Though also compatible with Arduino's IDE (Integrated Development Environment) in this guide we will be using Microsoft's IDE, download link follows:
<https://code.visualstudio.com/>

Optional but Handy:

- **Multimeter**- a useful tool for measuring continuity, DC voltage and resistance. While not essential for assembly, it's recommended to keep one to hand to identify any short circuits that may crop up during assembly. Also needed to test solar board voltage output once cells are fully connected.
- **De-soldering tools**- Such gizmos like spring loaded suckers, or material like copper solder wick can come in very handy, especially if you have no prior experience soldering. Soldering stations also can come with adjustable air flow re-work stations which can come in handy for removing damaged/misplaced components.
- **Scalpel or sharp craft knife**- always a handy thing to keep to hand.
- **Anti-Static Soldering Mat/Grounding Band**- When working with electrical components, it's wise to ground yourself to help avoid accidentally "Frying" any components with a static discharge. It's in the optional section but we'd definitely recommend it.

Extra Materials (our personal suggestion)

- **Blue Tack** - Very useful material when it comes to space age technology
- **Double Sided Tape** - Helps to position your sensor before soldering
- **Cuppa Tea** - Borderline essential for cognitive function, especially for the British!
- **Head Cleaner** - Helps keep your soldering tip clean and prevents sticking



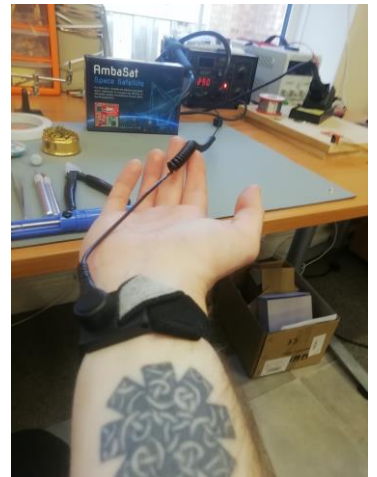
Health & Safety Precautions too

Though this is a fun and exciting project for anyone to undertake, this is far from a toy and must be respected. Where engineering is concerned, safety should always be considered no matter how experienced you are. If kids are involved, we would ask that an adult is present to observe and advise on the proper use of tools like a soldering Iron/station, hot air re-work tools and any sharp knives.

In this section, points of attention will be outlined in order to reduce the risk of injury and damage to yourself or your components.

What to Keep in Mind

STATIC! Yes, the components you are using are very sensitive to static charges and you should keep this in mind throughout the assembly process. Something as silly as wearing nylon socks on a cheap carpet could be enough to totally fry components on your PCB. What can we do to avoid this? Use an Anti-Static Soldering mat with a grounding band. With a band connected to something metal such as a desk leg, radiator, soldering station, or other... You can keep yourself grounded. This way you don't have to worry about static while you work. Also transporting your components or your assembled satellite using protective packaging/wrappings is recommended; the red bubble wrap provided with your kit is excellent for this. If anti-static mats or bands are not accessible, be sure to firmly grab hold of something metal before you start tinkering in order to discharge yourself.



VENTILATION! Indeed, it can be easily overlooked especially if you have no prior experience with soldering. Inhaling solder fumes is not recommended! A well-ventilated space is suggested for all soldering, an open window at least if proper extraction fans are not available. If, like me, your personal hobby cave is not well ventilated, a soldering station with a smoker scrubber is highly recommended.

HOT! No touchy, touchy! Yes! Believe it or not, the tip and shaft of your soldering iron can get extremely hot, along with the components you are heating too. Keep this in mind if using a rework air flow station too. Always turn off equipment such as soldering irons **Caution:** Tea is also hot! If it isn't, you'll need a fresh one.

WATCH THAT CABLE! A surprisingly easy way to mess up your day, unless you're using top end gear with silicone cables... your soldering iron can more than easily melt through your power cable's insulation. Keep in mind, where you're moving, placing, and resting your Iron to avoid this from happening.



Prep your Stations Cups of Tea at the ready!

Now that we have everything that we need, and we know what to keep an eye open for where health and safety is concerned, it's time to pour a cuppa tea and prep your work area. What needs prepping?

1. Tea, keep tea out of spillage range of your work area, but close enough to be at hand if needed.
2. Let's make sure you have a ventilated work space, open window if that's your best option.
3. Turn on your soldering iron, it may take a moment or two to reach operating temperature; if using a soldering station, adjust your temperature to best suit your solder.
4. Wet your Soldering Iron's sponge and/or make sure your Head Cleaner is close to hand (you don't want to go hunting for these things mid task)
5. Place Tools you may need out and ready, having tweezers at the ready to tinker in a hot area, or handle something delicate saves time if you keep them to hand. Same for your snips and any desoldering tools you may have.
6. GROUND YOURSELF! Either attach your grounding band to your wrist, or make your best efforts to discharge any built up static before moving to the next point.
7. Lay out your components in front of you, your PCB Motherboard, the Sensor Board, antenna, Pins and other materials you feel you may need.
8. Have your computer close with compatible IDE software ready to go



Don't Yet Have Visual Studio Code?

Though AmbaSat-1 is compatible with the Arduino IDE, for this guide we will be using Visual Studio Code. If you do not yet have this software please see the link below:

<https://code.visualstudio.com>

AmbaSat-1 also uses the PlatformIO extension alongside Visual Studio; this is a professional tool for system and software developers that will work across many platforms. It can be installed from Visual Studio Code directly by searching for “PlatformIO IDE” in the search bar found on the left of the window. Once the extension is found, click on the green “Install” Button.



Now we're on our way to having a usable IDE, you'll only need the AmbaSat-1 definition file (see below), and the most up to date source code.

The definition file for the AmbaSat-1 PCB holds the instructions for PlatformIO on how to configure your programmer gizmo. This file will need downloading from the following link:

<http://ambasat.com/downloads/AmbaSat-1.json.zip>

Once the Zip file is downloaded, unzip and copy the file “AmbaSat-1.json” to Platform IO's boards folder. This location may differ across devices. The following locations are good places to check if the location isn't obvious to you:

C:\Users<your username>.platformio\platforms\atmelavr\boards
C:\Users<username>.platformio\boards

If neither of these locations exist, then:

See these additional instructions at:

https://docs.platformio.org/en/latest/platforms/creating_board.html and specifically, you will need to create 'boards' directory in core_dir, if it doesn't exist.

core_dir location is:

Unix ~/.platformio

Windows %HOMEPATH%\platformio

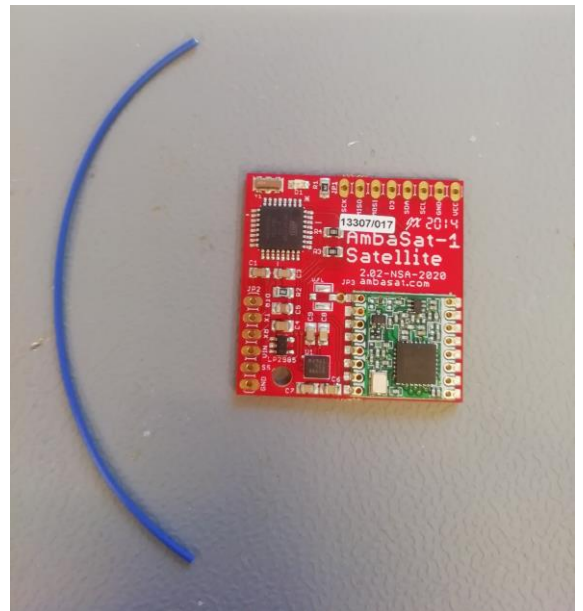
Okay, now that is all set up and ready to go, let's get hold of the most up to date source code for your satellite project. Because AmbaSat-1 is an open-source project, you can find all the code you'll need and plenty of other interesting pieces of information from the AmbaSat GitHub Repository. Follow the link below and hit the green "Code" button and select "Download Zip":

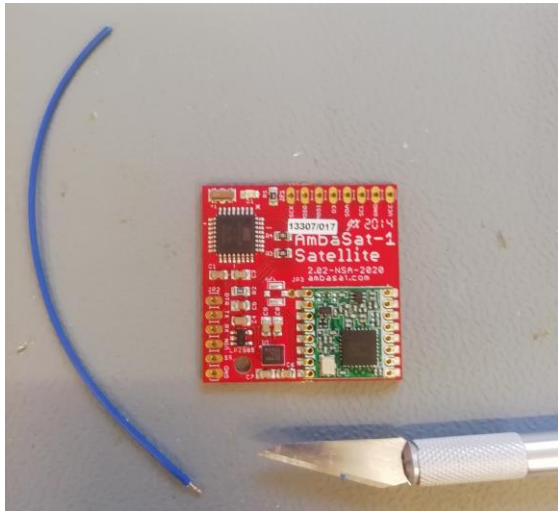
<https://github.com/ambasat/AmbaSat-1>

Step One- Slap a Tail on It!

It's important that this step is done before any testing of components. Why? When working with the AmbaSat-1 PCB, it's crucial not to power the board without the antenna attached, otherwise your satellite may apply power beyond safe limits and potentially frying the chip. This would require replacement parts in order to get full functionality again.

The length of your antenna needs to be 91.5mm to suit the frequency of 915mHz; this is because of what's known as the resonant frequency. If the cable is too short then not all of the signal will be transmitted but reflected back to your transmitter, damaging the transceiver. We provide one with a little extra length so please trim the antenna to the correct length.





Using your Snips, sharp knife or even wire strippers (we recommend using a knife to avoid taking off too much insulation), remove a couple of millimetres or so of the insulation from one end of the antenna.

Taking care not to have the antenna protruding too far through this hole (if at all) on the motherboard (**Circled in Blue in the below image**), and make sure not to short these points here (**Circled in Green**).

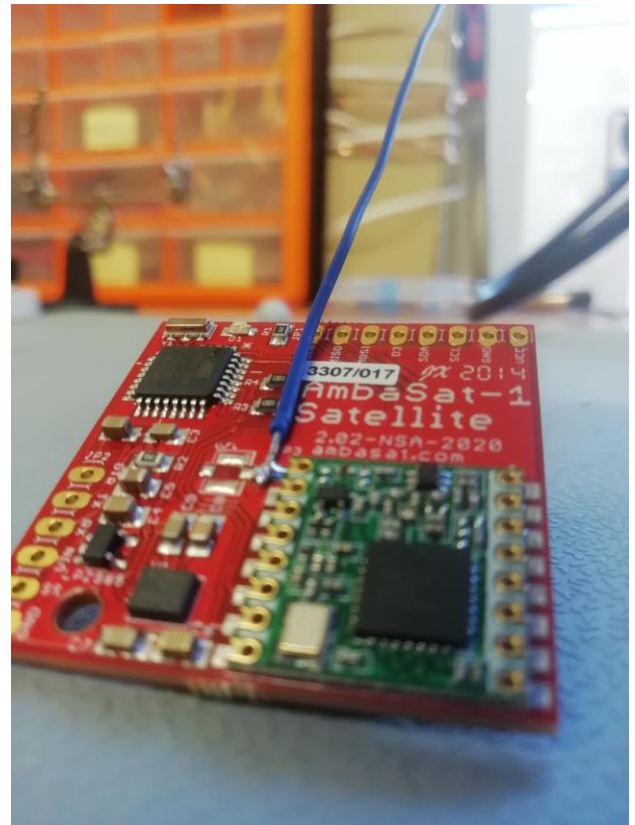
If any Points become shorted in the circled areas then this will stop your transceiver functioning. Shorts can sometimes be identified visually while other times with a multi-meter; they occur when you bridge two points that aren't meant to connect, creating a short circuit that power flows through. If any points accidentally become shorted by solder, heat up the solder till it starts to flow again, suck up the solder with your spring-loaded gizmo while it's still flowing; you could also place copper wick over the solder and heat with the iron (with tweezers to hold the wick, it will get hot!).



Attach the antenna in place, either resting slightly into the hole near your transceiver or between the two points either side of this hole, connecting with solder. If needed, any protruding wire can be cut flush with snips. Be wary of keeping your iron connected to the antenna for too long, insulation will start to melt rather quickly.

Blue Tack or tape may be used to hold the antenna in position while soldering, be careful not to cold solder a weak connection as the antenna will break free. If struggling to get in place; add solder in place first using your iron, get your iron and antenna ready, reflow the solder with the iron and then place the antenna in place.





Once secured in the desired place, cut down to 91.5mm length. Give a slight but firm tug on the antenna; make sure it stays secure because there's no fixing the loss of this once in orbit.

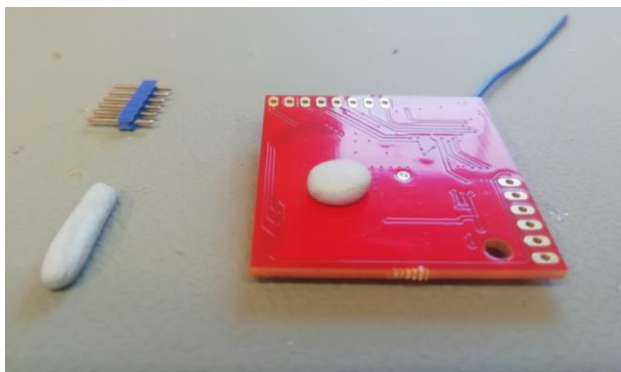
Done that? Good, let's keep going!

Please note: If you're AmbaSat-1 has a reserved slot onboard a rocket launch, we may replace your antenna prior to launch.

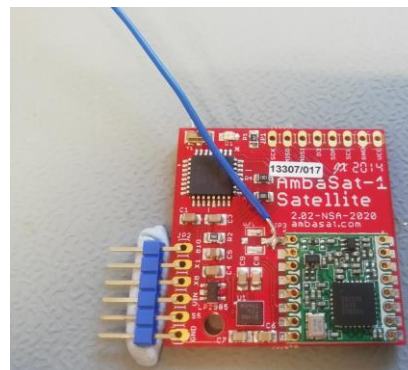
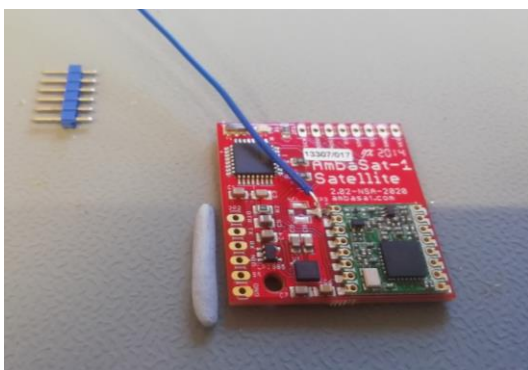
Step Two- Programming Pins

Now we have our antenna attached and at its proper length achieved, we can run power through the Ambasat-1 PCB. To do this we need to make use of our six programming pins. It's worth keeping in mind that these pins are only temporary and will be removed prior to launch and before the solar board is finally attached, and so we advise to use light soldering in order to just 'tack' the pins in place.

The connection points on the left side of this image are the points of contact for the programming pins. (This is where I personally like to start using Blue-Tack to make a hi-tech sausage to help keep things in place during the soldering). Line up your six pins with the "GND", "SS", "VIN", "RX", "TX" & "DTR".



As part of your Ambasat-1 Kit, you will be supplied with connection pins. These pins connect with the CH340 programmer to allow coding and testing as well as allowing your battery pack to connect to the assembled board for final testing. Again, keep in mind that these pins will need to be removed (which can be done by us here at AmbaSat once we receive your satellite prior to launch).



Once lined up, you'll just want to tack these points lightly.

I like to start with the "GND" point and the "DTR" to help attach them neatly.

You can attach the pins in whatever

order feels best for you, this is just the method I use to get the pins attached as neat as possible. Other methods are possible, and as long as the connection is made, the job will do. If too much solder is used, or you wish to remove these pins yourself after the final stages of coding... then either copper wick, or a spring-loaded gizmo sucker can be used. An airflow tool can also be used but be careful as this may overheat other components on the PCB!



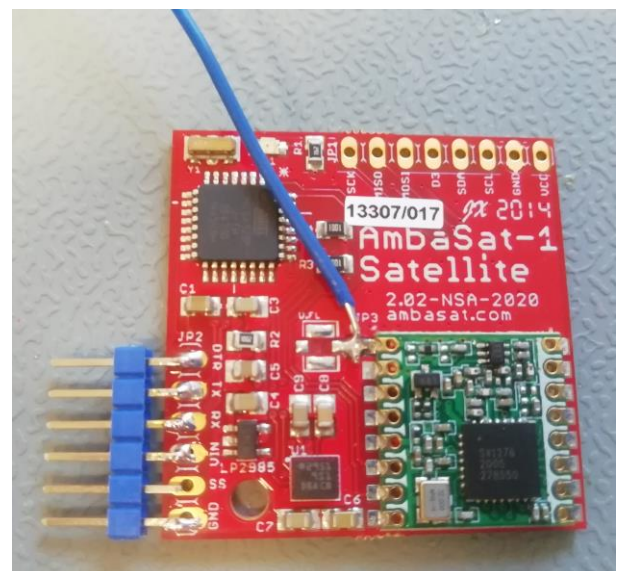
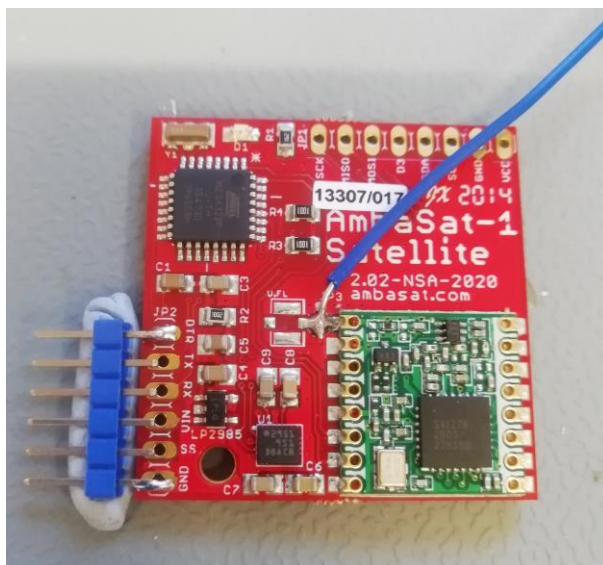
The “SS” connection point isn’t required here so to save work it’s recommended to keep this point unsoldered. Excessive heat isn’t needed so quickly glide your iron over the pin and to the connection point with just a bit of solder preloaded on to the tip of the iron in order to achieve a good connection.

Done? Good, you should have something that looks like the images below...

Now we can see where our programmer will be attaching, which means we’re at a really good point to test your PCB board and ensure it will accept your code. We will test by using Visual Studio Code and example code from the AmbaSat-1 GitHub repository.

The Ambasat-1 Motherboard has an LED fitted to the board to help show things are working as they should, but it’s not going to light up and flash on its own is it?

So, let’s make sure you have a nice cuppa tea to hand and move to the next step!

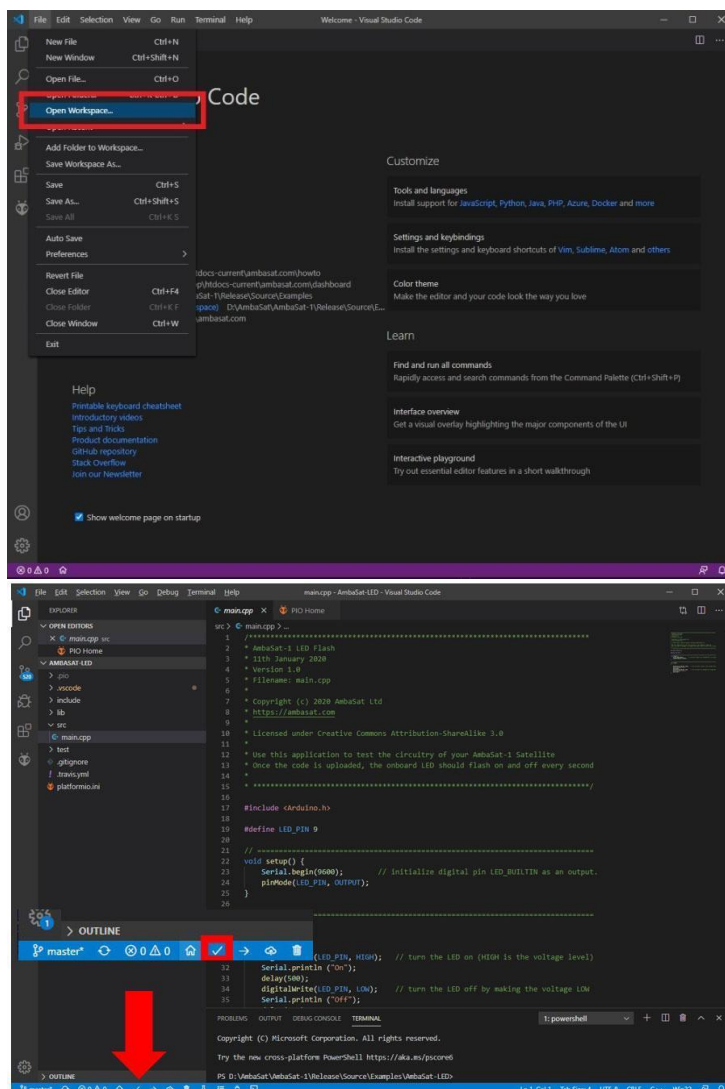


Step Three- LED test

Remember that source code ZIP file you downloaded from GitHub? Well, let's unzip the file to a folder on your computer. We suggest something simple like: **C:/ambasat**



Now it's time to build our "digital workspace". This step will help test your AmbaSat-1 board and chosen sensor. Let's start with the LED code as this will give us a good indication that everything is working fine and also a good first look at using the example code. Make sure you have internet connection for this step.



In *Visual Studio Code*, click on file and find the option "Open Workspace..." Then find your AmbaSat folder where you have unzipped your code. Look for the "AmbaSat-LED" example file that's should be located at: AmbaSat-1>Release>Source>Examples

Once open in Visual Studios Code, any library files which are needed by the project will be automatically downloaded. This may take a short while before building or editing can be dabbled with. Once loaded it's time to build the LED program.

To build, click on the small tick on the bottom bar of your Visual Studios Code window, as indicated on the image with the big red pointer.

Building puts together the code into something usable by your AmbaSat-1. You will see the framework compiling when you do this, and if the build was successful you'll be shown a green "Success" notification after the compiling is complete.

Now let's get connected!

With the Programmer provided in your Kit, find and attach the wires to the programmer's connection pins, they should slide on with minimum effort. You'll notice each pin is labelled similarly to your PCB with markings for "DTR", "RXD", "TXD", "VCC", "CTS" & "GND" and these connect to the PCB pins as follows:

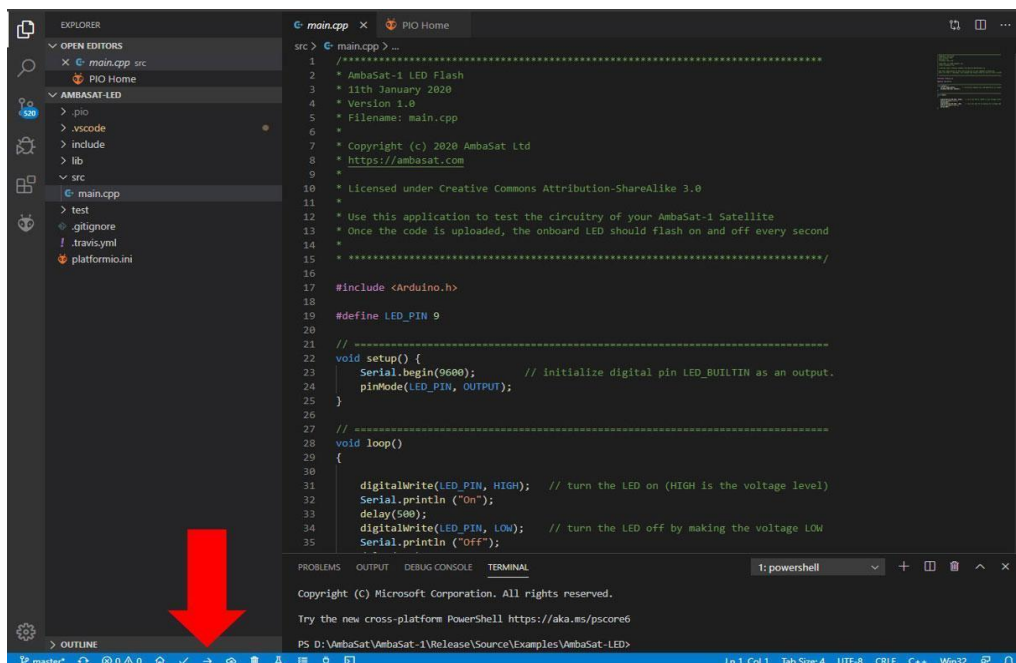


G - G
VCC - VIN
RXD - TX
TXD - RX
DTR - DTR



Again, keep in mind how sensitive your components are to static and electrical charge, stay grounded if and where possible while connecting the AmbaSat-1 PCB to the programmer. Once connected, plug the programmer into your computer's USB port.

Now with your Built LED workspace, your programmer attached to your PCB and computer we need to copy the code across to the satellite. To copy across your build, click the small arrow located next to the build button on the bottom bar of your Visual Studio Code window. You will see the progress of the copying over in the window along with any errors that may occur during this process (we will cover some of the more common errors in the next section).



Once copying is complete, you should see a line of text saying "SUCCESS" in the Visual Studio Code window. Now take a look at your satellite... See that flashy green LED thing? Well done! This is a good test for our PCB as it shows us your AmbaSat-1 satellite is accepting programming and power is flowing through the circuitry correctly. This is also a good first exercise in seeing the cause and effect of simple code.

Error? Error! What Does it Mean?

When working with code, errors are bound to happen, in this section we'll briefly cover some of the more common errors you may come across during your AmbaSat project, together with possible solutions to overcome them. When unsure, remember the IT Crowd words of wisdom "Hello... Have you tried turning it off and on again?"

Error: Please Specify 'Upload Port'

This is an easy problem to fix but one which is probably a more common error than you'd think.

```
LDF Modes: Finder ~ chain, Compatibility ~ soft
Found 8 compatible libraries
Scanning dependencies...
No dependencies
Building in release mode
Checking size .pio\build\AmbaSat-1\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM:   [=         ]   9.5% (used 194 bytes from 2048 bytes)
Flash: [=         ]   6.1% (used 1980 bytes from 32256 bytes)
Configuring upload protocol...
AVAILABLE: arduino
CURRENT: upload_protocol = arduino
Looking for upload port...
Error: Please specify 'upload_port' for environment or use global '--upload-port' option.
For some development platforms it can be a USB flash drive (i.e. /media/<user>/<device name>)
*** [upload] Explicit exit, status 1
===== [FAILED] Took 1.54 seconds =====
The terminal process "C:\Users\Nathan Walls\.platformio\penv\Scripts\pio.exe 'run', '--target', 'upload'" terminated with exit code: 1.

Terminal will be reused by tasks, press any key to close it.
```

Solution A:

Check that you have not forgotten to plug your USB programmer in (it happens)

Solution B:

Change USB ports, it's possible your USB port is damaged and a swap should resolve this

*** [Upload] Error 1- avrdude: stk500_recv(): programmer not responding...

This is another common error but usually easy enough to fix

Solution A:

Check that your programmer wires are fully connected and not sliding off the connection pins on either the PCB or the programmer itself.

Solution B:

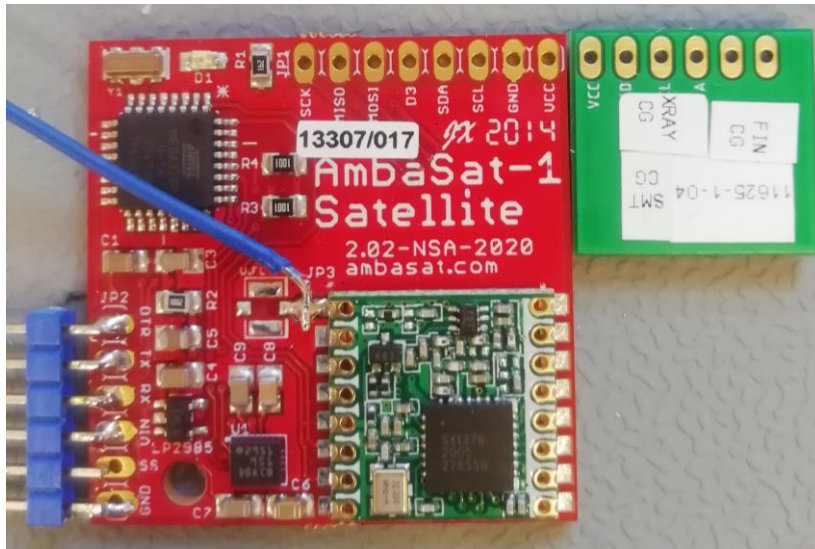
Check that your PCB is connected to your programmer as shown in the previous section. It's easy to miss one wire or accidentally connect the wrong pin to the wrong wire.

Solution C:

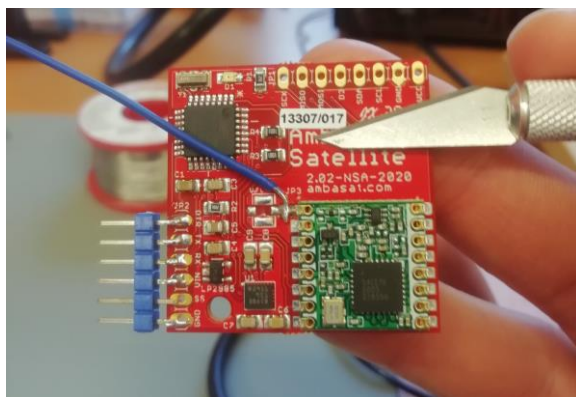
You may have a loose connection where your pins have been tacked to your PCB. This can be sometimes remedied by simply running your soldering iron back over the pins and to the connection pad; other times a small amount of extra solder is enough to create a better connection.

Step Four- Mount your Sensor

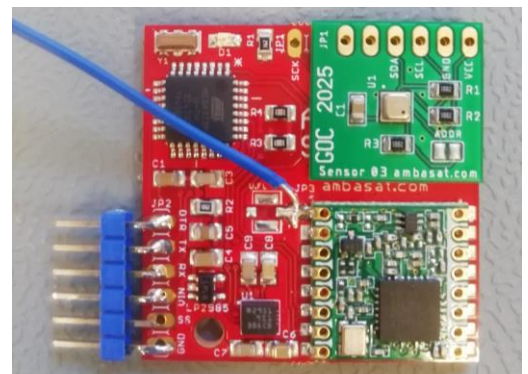
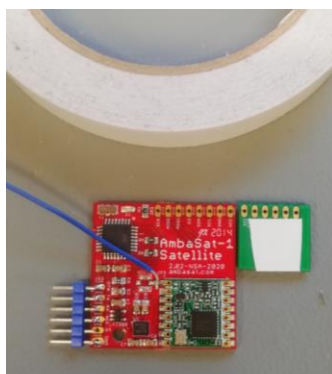
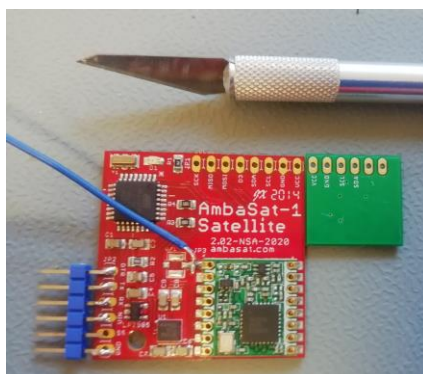
Now it's time to get your chosen sensor mounted. There are a few different approaches to this stage but I will go through the best way I have found that works for a flush solder on the rear of your PCB. Why is this important? Your solarboard will later mount onto the rear of your PCB. Not only is the board delicate, we also need to reduce the risk of gaps between the two boards as this would allow the solarcells to flex and potentially break. There's only so much room in the 3U Launch container and every millimetre counts. So, let's be as neat, and tight, as we can.



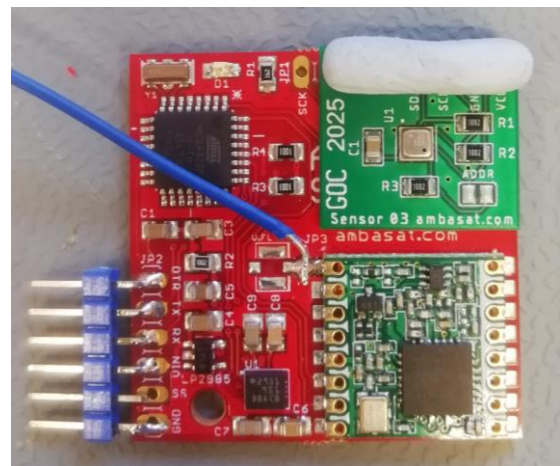
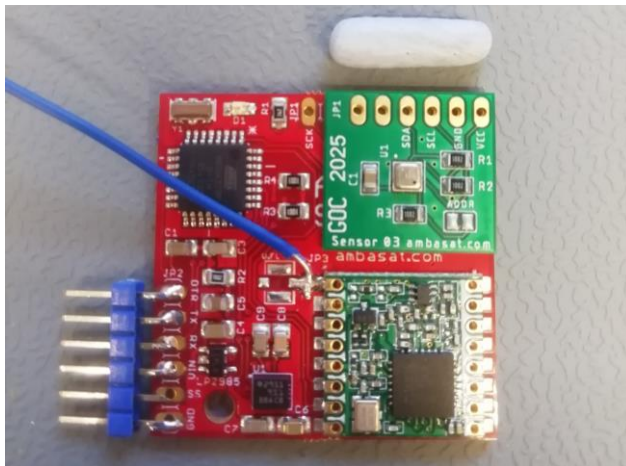
In the spirit of every millimetre counts, check your AmbaSat-1 PCB and also the rear of your sensor board for any stickers that may stop your sensor sitting flush with the main PCB. This is where a scalpel or a sharp craft knife comes in handy. Remove any stickers to get mounting as tight as possible.



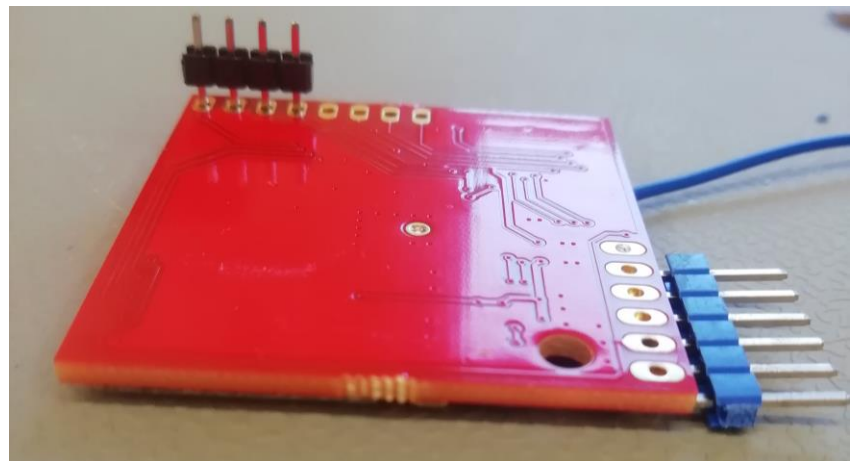
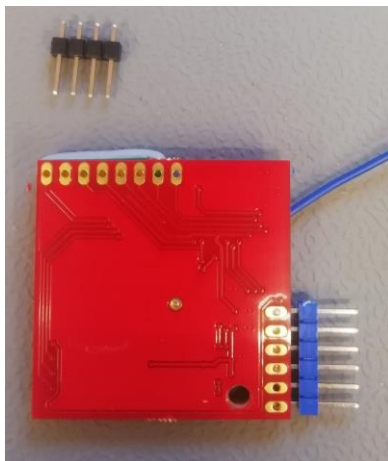
Once the stickers have been removed, use a square of double-sided tape (or light amount of appropriate adhesive... maybe not chewing gum) and roughly cover the rear of your sensor. Be sure to leave the connection pads clear! Stick your sensor squarely across the top right corner of your PCB so the last four connection points line up as perfectly as you can. Your satellite should look something like the images below.



Now that you have your sensor in place, it's time to solder it into position. In your kit's bag of components you will find extra pins. (You could also use different materials such as resistor wire but the pins work just fine for this). With this method we will solder from the rear first as gravity will pull the solder down and through the connection point and we want a flush finish on the rear, not a conical point like you often see where solder is used. Take four pins, and a small amount of Blue Tack if you want to be hi-Tech; I like to roll out a small Blue Tack sausage the same length as the sensor, take the Blue Tack and place it over the top of the sensor connection points.



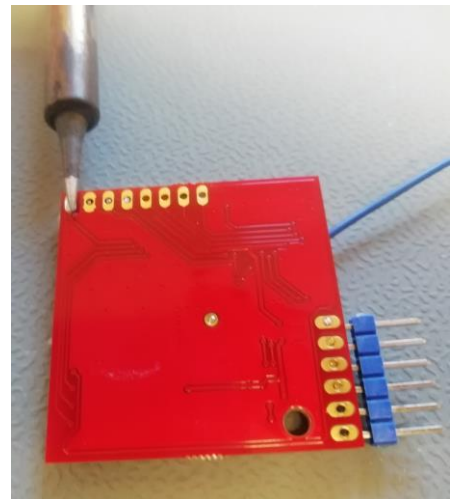
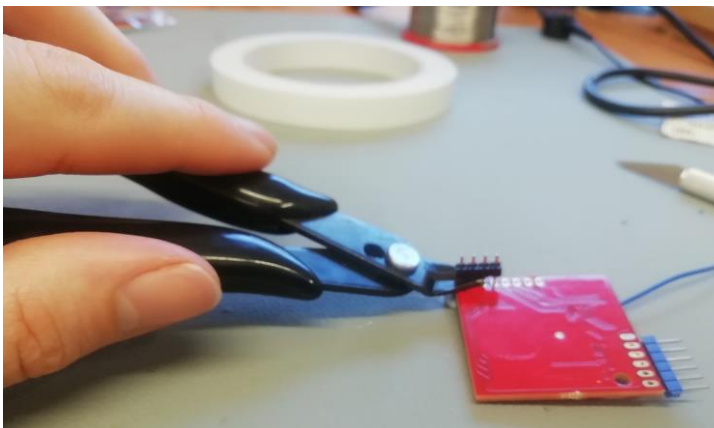
Now turn the board over so your satellite is facing downward. Lightly apply a small amount of pressure to the corner above your blue tack sausage (**caution:** do not eat the sausage). With your four remaining pins, remove them from the plastic binding (or remove later with snips) and feed them into the holes of the connection points of both the sensor and the PCB until you feel them connect with your Hi-Tech Sausage.



Once you're happy that all pins are through, use some sharp snips or pliers to cut the pins as close to the board as possible to form a flush finish. If any pins are still protruding through the rear, gently tap them deeper into the Hi-Tech Sausage until they sit flush or slightly embedded.

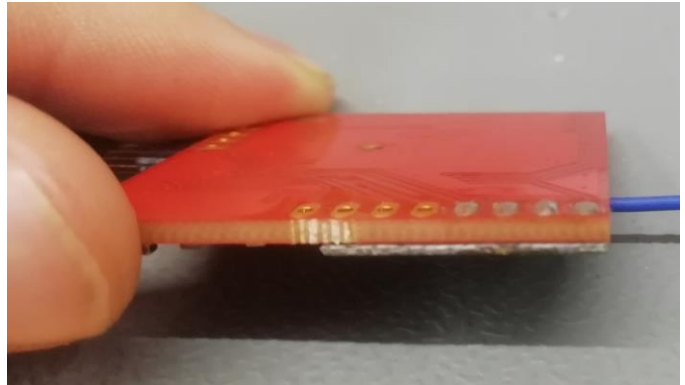
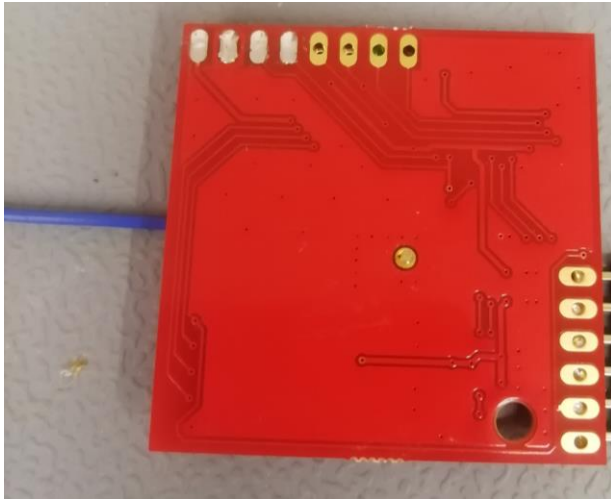
Now that you have the pins cut down at the rear, it's time to solder them in place. Using your soldering iron, heat the connection points one at a time for a second or two before feeding in a small amount of solder. You'll want to feed the solder on to the heated connection pad and into the hole, not feeding directly on to the iron.

Less is more and we don't need much solder in place. Don't forget, more solder may trickle though once we solder the top side of the sensor board.

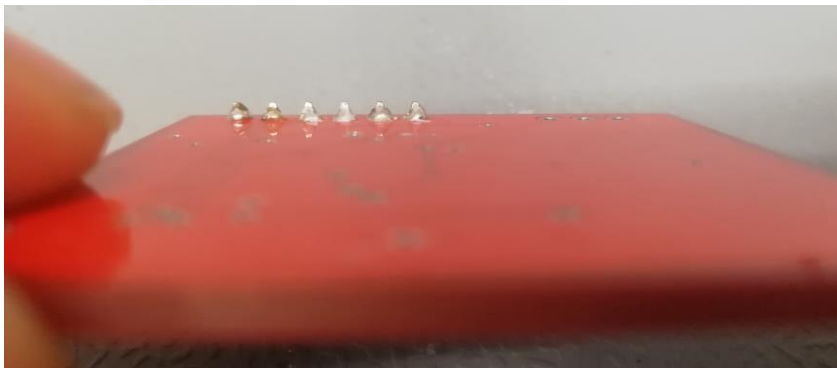


Once the desired amount of solder has been added, remove the solder wire and briefly hold the iron in place for a moment or two before gently brushing away from the connection point. This should leave you with a clean flush solder. Clean the tip with your sponge or head cleaner and move to the next connection point (this stops your iron from sticking). Repeat this for the next 3 connection points.

Check to make sure you haven't applied too much solder, if the rear is looking a little bumpy, you may want to run over it with copper wick or use the spring-loaded sucker gizmo to clean up the area.

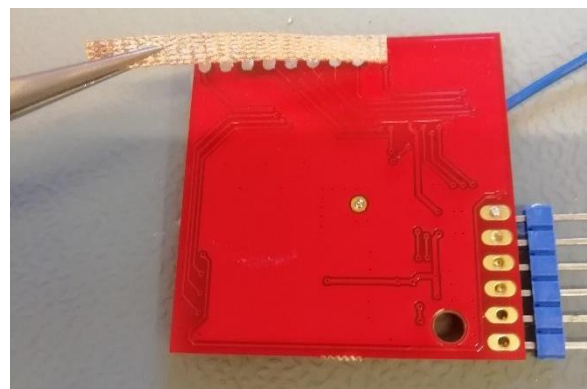


Your solder should look nice, smooth and flush... no bumps or conical tips like the next image.

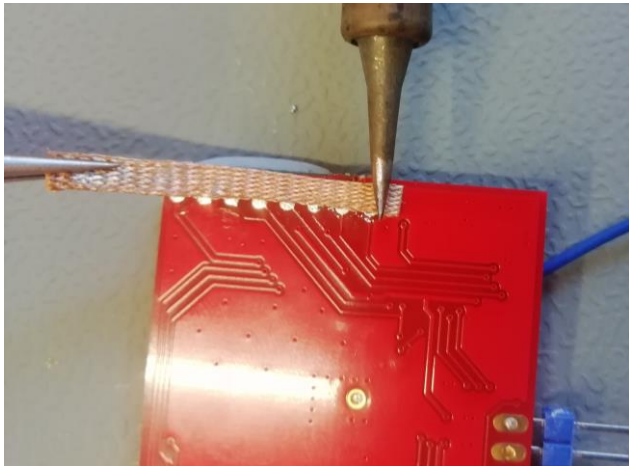


This is an example of what you **do not** want your solder to look like! I will show an example of de-soldering using copper wick; this is useful for removing any accidental solder runs, cold solders, or excess solder.

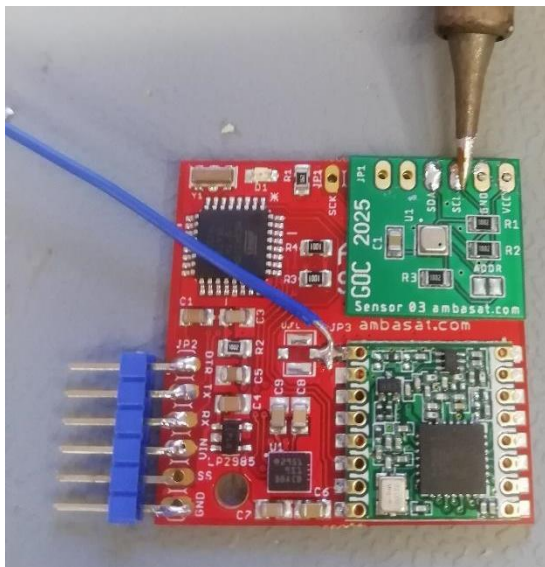
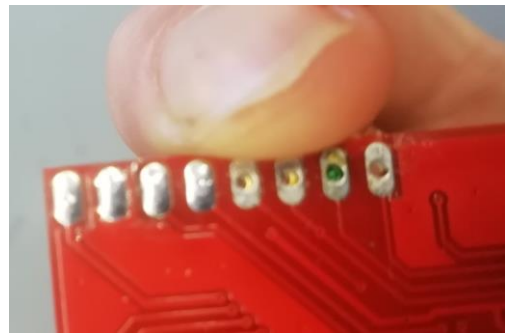
Let's say you accidentally solder too many connections, how would copper wick help in this situation? Place copper wick over the solder and heat with the iron (with tweezers to hold the wick, it will get hot!).



As you notice the solder begin to flow into the wick, keep the wick moving so as not to get it stuck. Heat each point as needed to remove the solder. The spring-loaded sucker may also be used here. Simply reflow the solder with your iron and place the sucker over the flowing solder and hit the button. So, if you make mistakes, stay calm; if you take your time you can remedy them and keep on going with your project and be wiser for it.

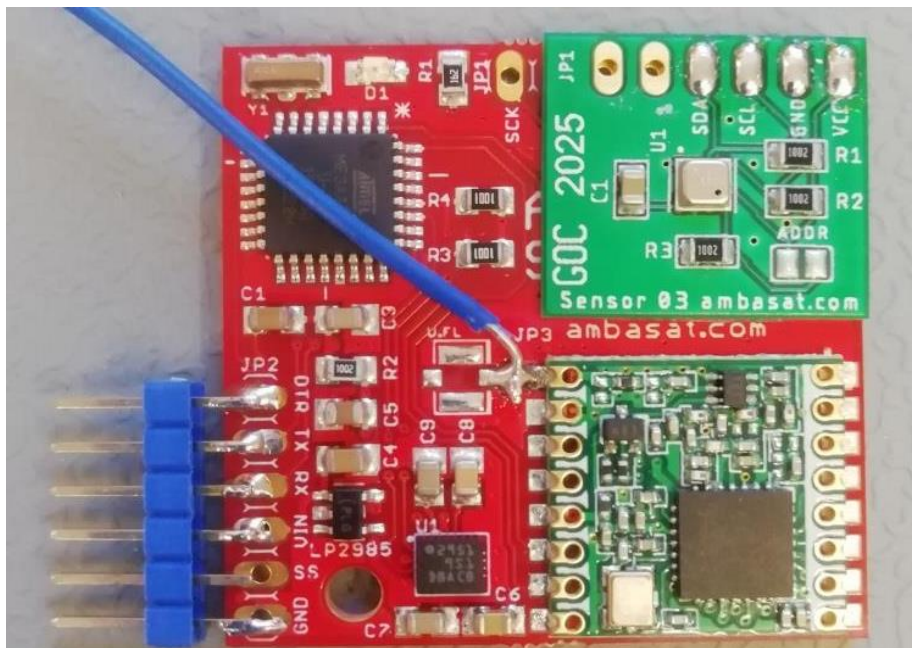


But back to mounting your sensor!



With all four connection points soldered at the rear it's time to flip it over and remove your Hi-Tech sausage. Having a conical finish to your soldering at this side of your sensor isn't an issue like it is with the rear, so just roughly cut down your pins at this side and solder in a similar fashion. Heat the pad and pin for a couple of seconds, feed in the solder, keep the iron in place for a second or two longer than the solder wire and gently lift upwards to have a conical finish. Do this on each of the four points and you should have something that looks like the image below.

Once the top side of your sensor is soldered, recheck the soldering at the rear of your PCB. Sometimes excess solder runs down and through the connection points, ruining your flush solder. Sometimes this can be solved by re-flowing your solder, run your iron over the rear connections while your satellite is faced down and allowing the solder to flow for a moment; sometimes this is enough to encourage the solder to run smooth and flush again. Other times you may have to use either Copper Wick or a Gizmo Sucker to either adjust or redo the solder.

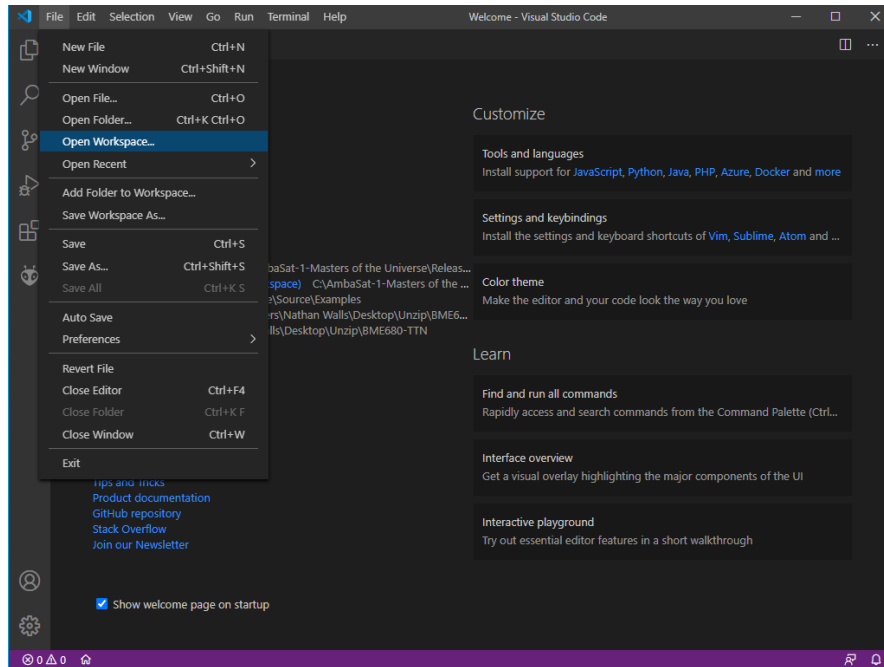


Congratulations! Your chosen sensor is now properly mounted to your Satellite's motherboard, very exciting! But let's not rush into getting it launched just yet... We next need to make sure everything is communicating and functioning correctly.

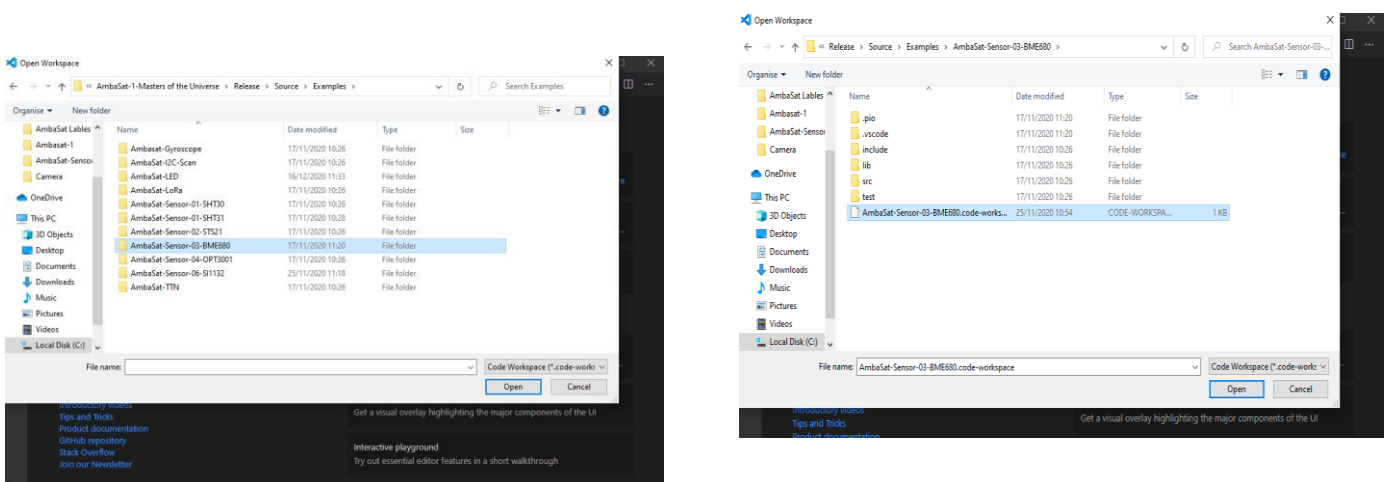
Step Five- Is My Sensor Sensing?

Similar to the LED test, we'll now go over a quick way of testing your sensor. The sensor you have chosen will have a Visual Studio 'workspace' amongst the source code files we downloaded earlier. Open Visual Studio Code.

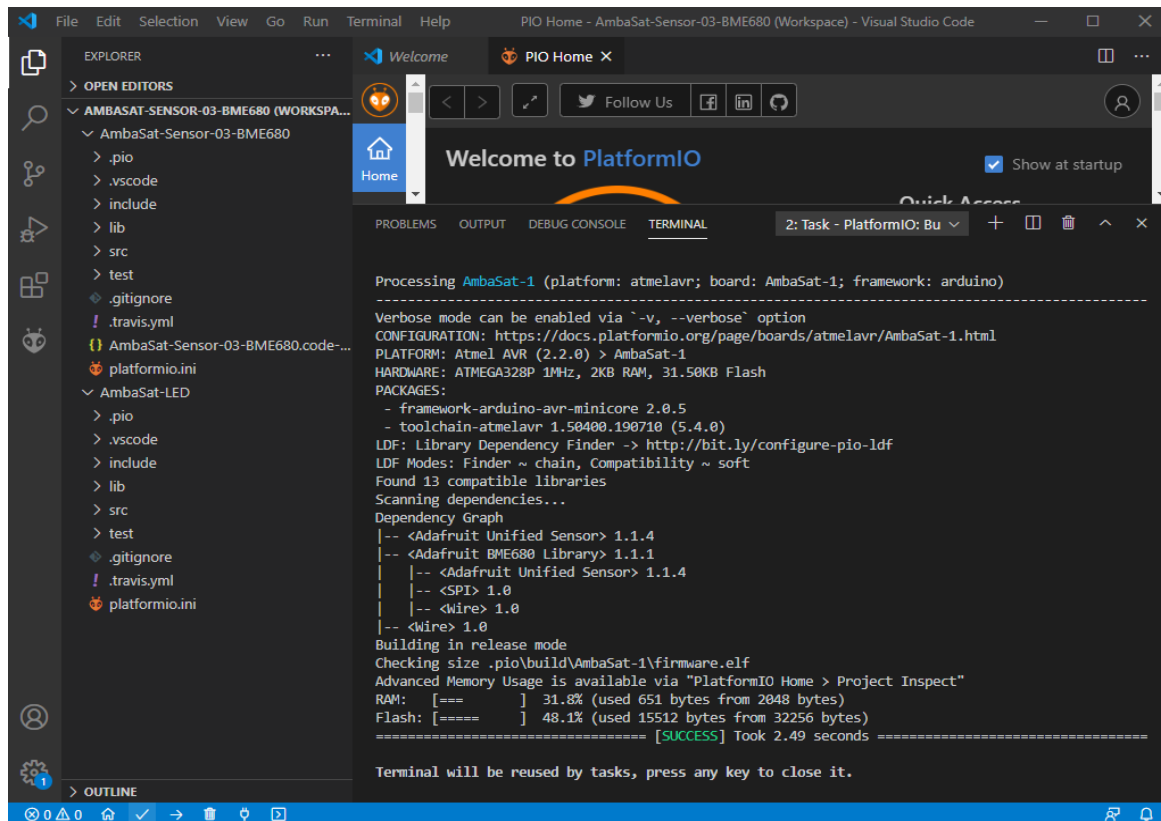
Go to file- Open workspace:



Under "examples", locate the folder applicable to your chosen sensor, inside you'll find the digital workspace needed to test your sensor:



Once loaded, in the same way as you did earlier, hit the build button (little tick on the bottom bar). Once the build is complete, you'll get a screen similar to the image below and a chunky green "SUCCESS".



```

PIO Home - AmbaSat-Sensor-03-BME680 (Workspace) - Visual Studio Code
Welcome
PIO Home X
Home
Welcome to PlatformIO
Show at startup
2: Task - PlatformIO: Bu
Processing AmbaSat-1 (platform: atmelavr; board: AmbaSat-1; framework: arduino)
-----
Verbose mode can be enabled via `-v, --verbose` option
CONFIGURATION: https://docs.platformio.org/page/boards/atmelavr/AmbaSat-1.html
PLATFORM: Atmel AVR (2.2.0) > AmbaSat-1
HARDWARE: ATMEGA328P 1MHz, 2KB RAM, 31.50KB Flash
PACKAGES:
- framework-arduino-avr-minicore 2.0.5
- toolchain-atmelavr 1.50400.190710 (5.4.0)
LDF: Library Dependency Finder -> http://bit.ly/configure-pio-ldf
LDF Modes: Finder ~ chain, Compatibility ~ soft
Found 13 compatible libraries
Scanning dependencies...
Dependency Graph
|-- <Adafruit Unified Sensor> 1.1.4
|-- <Adafruit BME680 Library> 1.1.1
|   |-- <Adafruit Unified Sensor> 1.1.4
|   |-- <SPI> 1.0
|   |-- <Wire> 1.0
|-- <Wire> 1.0
Building in release mode
Checking size .pio\build\AmbaSat-1\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM: [====] 31.8% (used 651 bytes from 2048 bytes)
Flash: [=====] 48.1% (used 15512 bytes from 32256 bytes)
===== [SUCCESS] Took 2.49 seconds =====
Terminal will be reused by tasks, press any key to close it.

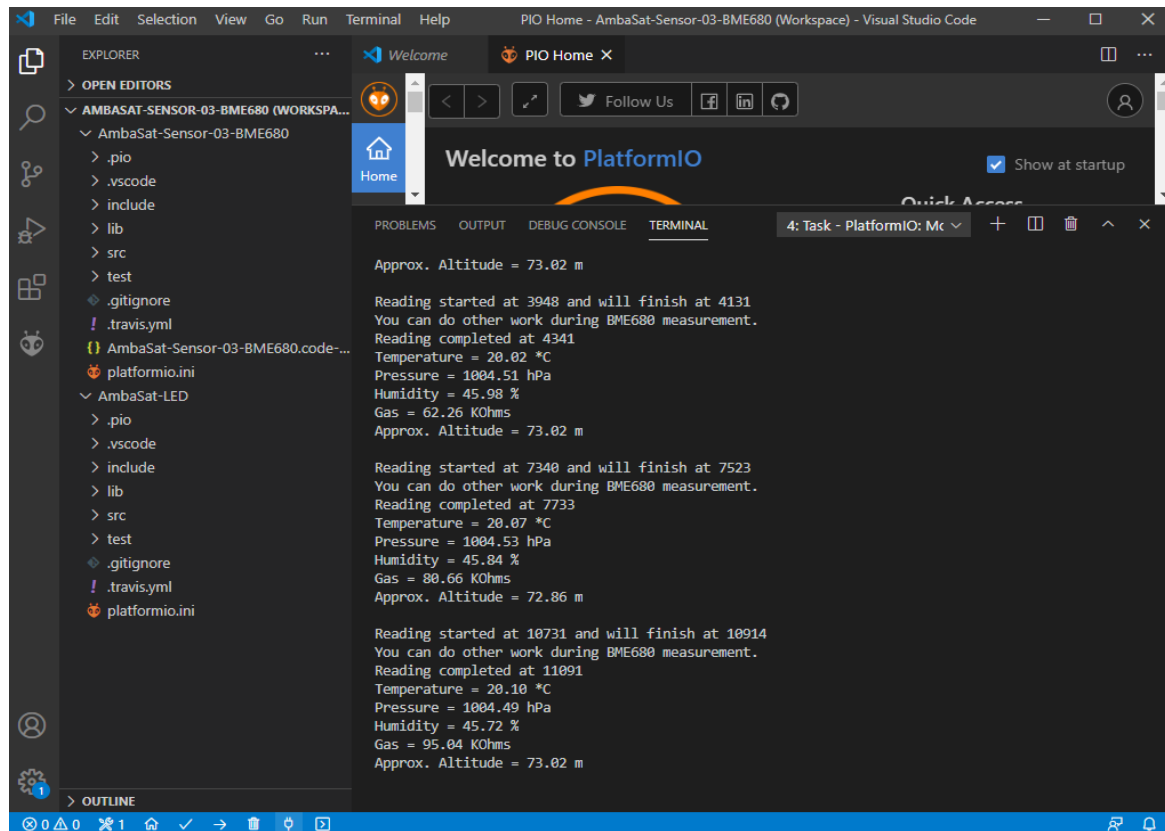
```

In the same way as before, connect your satellite to the programmer, and then the programmer to your computer. Make sure wires are connected in the correct sequence and remember to be wary of static, keep yourself grounded wherever possible.

Copy this build across to your satellite by clicking the small arrow near the "build" button. Don't worry if this stops the LED from flashing, there's only limited memory available after all.

Once copied across we will take a look at our terminal window. This will give us a direct look at the data your sensor is picking up.

The terminal can be accessed by the plug shaped Icon on the bottom bar of Visual Studio Code and will display the data that will be fed to the AmbaSat Dashboard once we link your device in the next step. As you can see, sensor 03 is giving us the rundown of the current conditions in the AmbaSat office, this shows it works



```
Approx. Altitude = 73.02 m

Reading started at 3948 and will finish at 4131
You can do other work during BME680 measurement.
Reading completed at 4341
Temperature = 20.02 *C
Pressure = 1004.51 hPa
Humidity = 45.98 %
Gas = 62.26 KOHms
Approx. Altitude = 73.02 m

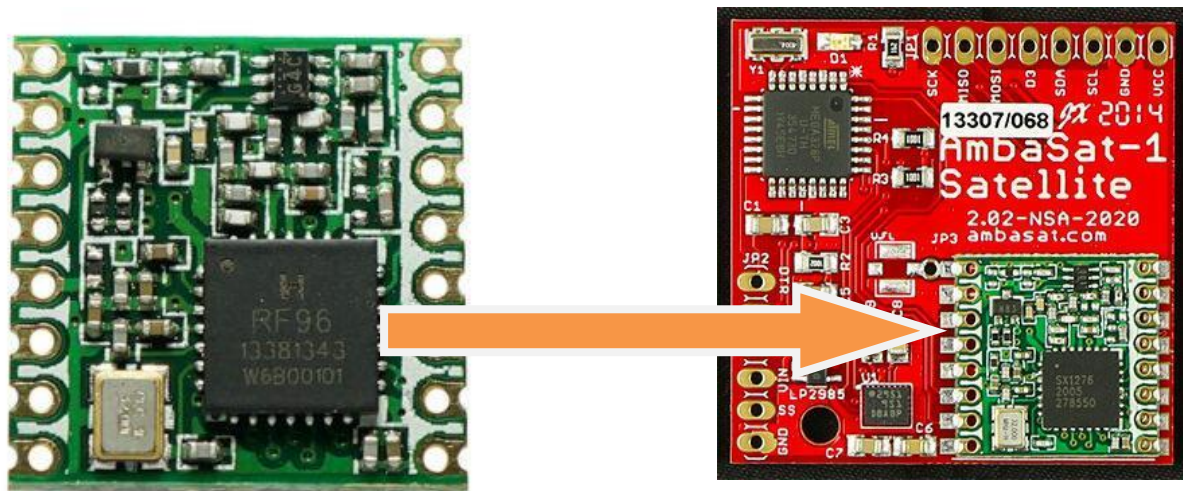
Reading started at 7340 and will finish at 7523
You can do other work during BME680 measurement.
Reading completed at 7733
Temperature = 20.07 *C
Pressure = 1004.53 hPa
Humidity = 45.84 %
Gas = 80.66 KOHms
Approx. Altitude = 72.86 m

Reading started at 10731 and will finish at 10914
You can do other work during BME680 measurement.
Reading completed at 11091
Temperature = 20.10 *C
Pressure = 1004.49 hPa
Humidity = 45.72 %
Gas = 95.04 KOHms
Approx. Altitude = 73.02 m
```

So, we have a working satellite board, a functioning sensor mounted... But what good is that if we can only read data while it's plugged in? This is where TTN and LoRaWAN comes into play!

Step Six- That's LoRaWAN!

What is LoRaWAN? LoRaWAN stands for Long Range Wide Area Network; it uses sub-gigahertz radio frequencies which have a proven functional distance of 700km+ (line of sight) so more than enough for our target of 310km in Low Earth Orbit. Luckily, with the AmbaSat-1 kit, you have a LoRaWAN transceiver pre-soldered to your PCB!



So ok, we know how our little Sprite Satellites are going to be talking to us, but how are we going to listen? This is where the non-profit organisation TTN (The Things Network) comes into play.

What is TTN? Well in their own words:

"The Things Network is about enabling low power Devices to use long range Gateways to connect to an open-source, decentralized Network to exchange data with Applications."

They do this brilliantly with the aid of people who freely install gateways worldwide to aid in their own or other's tinkering. Currently there are over 16,600 gateways worldwide and growing day by day.

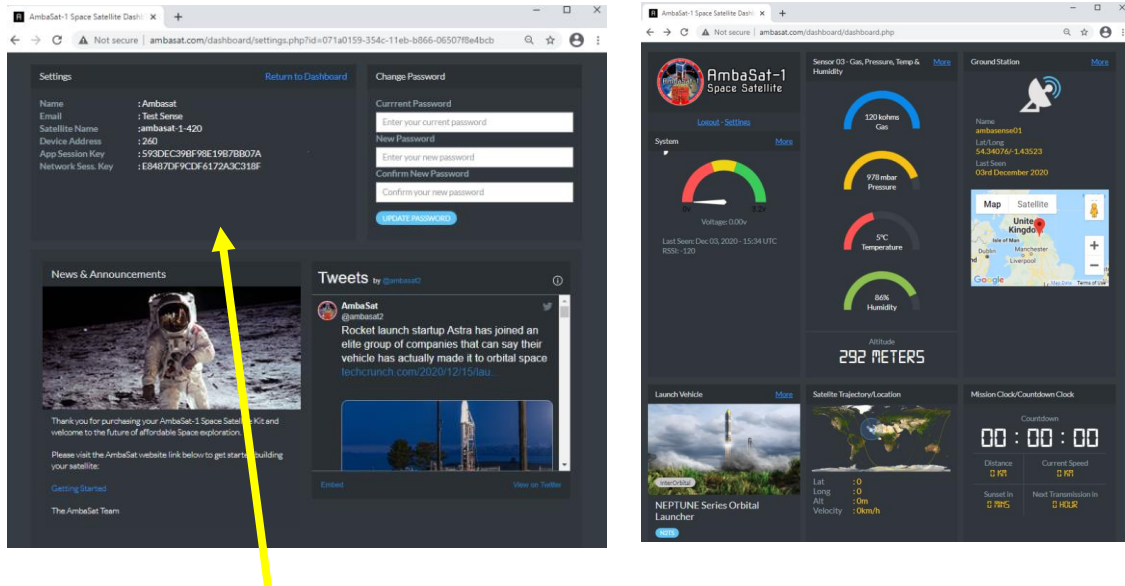
Check them out with the link below.

<https://www.thethingsnetwork.org>

So how do we make use of this technology? Back at AmbaSat HQ, we pre-register your device on TTN and provide you three important parameters through your AmbaSat dashboard account (you should receive log in details via email shortly after shipping). So, what information do we need to access and where do we find it? You will need:

1. NWKSKY - Network Session Key
2. APPSKY - LoRaWAN AppSKY, the application session key
3. DEVADDR - LoRaWAN end-device address

To find these, log in to the AmbaSat Dashboard and select the link to settings.



Here you can see an example of where you should be looking. These will now need entering into your code so that your AmbaSat-1 is correctly identified and knows who gets the data it is reading.

With your satellite still connected via your programmer, close your terminal and open a new Visual Studio Code window. In this new window, open the “AmbaSat-TTN ‘workspace’” found in your AmbaSat folder, then open the “main.h” file; which should be found around here: Release>Source>Examples>AmbaSat-TTN>src. Once the main.h file is open in the work space you’ll see a screen like this:

```

C main.h x PIO Home
BME680-TTN > src > C main.h > [N] NWKSEKEY
1  /* *****
2  * AmbaSat-1 Release
3  * Filename: main.h
4  * LoRaWAN encrypted data submission to TTN
5  * 1st August 2020
6  * Version 1.01
7  * Filename: main.h
8  * Authors: Martin Platt
9  *
10 * Copyright (c) 2020 AmbaSat Ltd
11 * https://ambasat.com
12 *
13 * licensed under Creative Commons Attribution-ShareAlike 3.0
14 * *****
15 *
16 * This example will send data to the AmbaSat Dashboard
17 * using frequency and encryption settings matching those of
18 * the The Things Network.
19 *
20 * This example uses ABP (Activation-by-personalisation), where a DevAddr and
21 * Session keys are preconfigured (unlike OTAA, where a DevEUI and
22 * application key is configured, while the DevAddr and session keys are
23 * assigned/generated in the over-the-air-activation procedure).
24 *
25 * Note: LoRaWAN per sub-band duty-cycle limitation is enforced (1% in
26 * g1, 0.1% in g2), but not the TTN fair usage policy.
27 *
28 * To use this code, first register your application and device with
29 * the things network, to set or generate an AppEUI, DevEUI and AppKey.

```

Here in this borderline “Matrix”-style file, you will get a rundown of what your code does and how everything works. Congratulations, it’s now time to get creative, modify and edit code and get your satellite to do its thing.

Make sure you have your Network Session Key, App Session key and Device address to hand for this next step (or temporarily copy and paste them into the file so you don’t have to keep switching windows).

Scroll down in your Visual Studio Code window until you see lines of text similar to the image below, this is where you'll input your unique keys and device address.

```

C main.h x PIO Home
BME680-TTN > src > C main.h > ...
96 // TTN *****
97 // The Network Session Key
98 static const PROGMEM u1_t NMSKEY[16] = {0xE8,0x48,0x7D,0xF9,0xCD,0xF6,0x17,0x2A,0x3C,0x31,0x8F,0x26,0x61,0x77,0x9D,0xEF};
99
100 // LoRaWAN AppSKey, application session key
101 static const u1_t PROGMEM APPSKEY[16] = {0x59,0x3D,0xEC,0x39,0xBF,0x98,0xE1,0x9B,0x7B,0xB0,0x7A,0xED,0x1F,0xA7,0x88,0x00};
102
103 // LoRaWAN end-device address (DevAddr)
104 static const u4_t DEVADDR = 0x2601139E ;
105 /*****
106
107 // These callbacks are only used in over-the-air activation, so they are
108 // left empty here (cannot be left out completely unless
109 // DISABLE_JOIN is set in config.h, otherwise the linker will complain).
110 void os_getArtEui (u1_t* buf) { }
111 void os_getDevEui (u1_t* buf) { }
112 void os_getDevKey (u1_t* buf) { }
113
114 static osjob_t sendjob;
115 static osjob_t initjob;
116
117 // AmbaSat lmic pin mapping
118 const lmic_pinmap lmic_pins = {
119     .nss = 10,
120     .rxtx = LMIC_UNUSED_PIN,
121     .rst = 7,
122     .dio = {2, A2, LMIC_UNUSED_PIN},
123 };
124

```

With your keys and address to hand, replace the values with your own two digits at a time, ahead of each “0x”.

For example, if you Network session key was:

5A E0 69 0C 7E E4 F4 A9 73 88 E2 1C A7 5B FDB1

You would change these highlighted values in this line:

{0xE8,0x48,0x7D,0xF9,0xCD,0xF6,0x17,0x2A,0x3C,0x31,0x8F,0x26,0x61,0x77,0x9D,0xEF};

To match the values of your unique Network session key like so:

{0x5A,0xE0,0x69,0x0C,0x7E,0xE4,0xF4,0xA9,0x73,0x88,0xE2,0x1C,0xA7,0x5B,0xFD,0xB1};

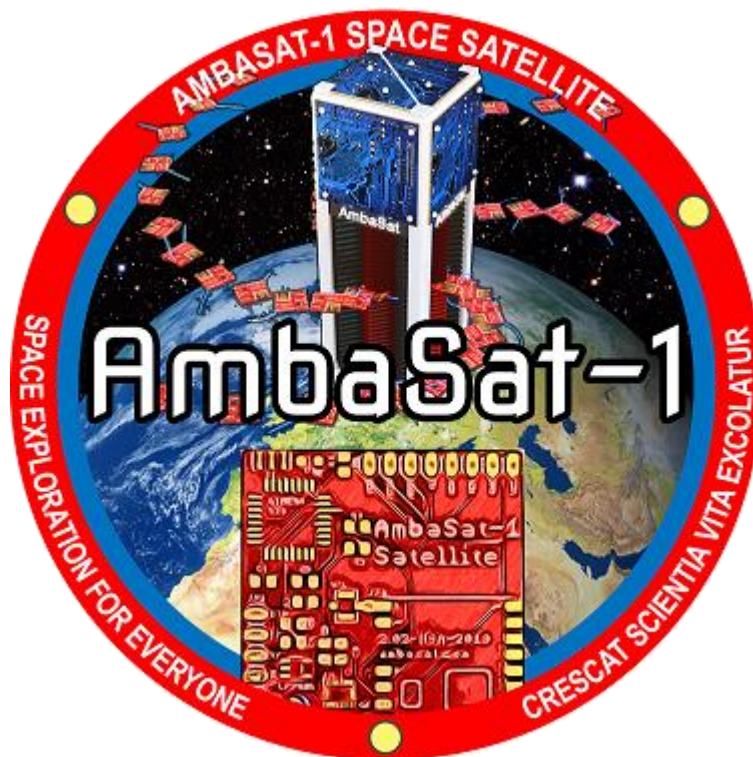
Double check your values are correct, maybe triple check if you’re anything like me. Once you are sure you have your values input correctly, hit the build button to compile your code... All good? You should see another “SUCCESS” at the bottom of your window. Now with your code compiled, hit the arrow button on the bottom bar of the window to throw it across to your satellite. “SUCCESS”? Well done.

You can now safely disconnect your programmer from the computer and the programming pins on your AmbaSat-1 Satellite.

Finally, in this “How To Guide”, connect your battery pack to the satellite (Red to VIN and Black to GND) and log in to your AmbaSat Dashboard account to see your data being received to your own personal ground station!

Congratulations and well done!

You have just taken the first steps of an exciting journey into Low Earth Orbit and beyond.



AmbaSat-1 Satellite Support

If you experience any issues or require any support, please use the following resources:

AmbaSat-1 Forum:

<https://ambasat.com/forum3/public/>

Email:

support@ambasat.com