Apricot Belgian Blonde 10 Gallon Recipe // All Grain

brew-day: 12/26/2015 // OG – 1.072 // Update on temp. chart of fermentation later along with FG, we are hoping for 8%. This recipe originally called for 3lb of sugar, we used 1.5lb. If you want 9%, add that extra sugar.

For yeast using a 2000ml starter, French Saison #3711 prepared 28 hours ahead of time. The best thing to do with starters if you really want to be exact about it, is to test the wort until it reaches an OG of approx. 1.040. This will prepare and propagate the yeast for the main fermentation without tiring its self out before the main fight. The best way to do that is with a refractometer (make sure to buy one with the SG wort scale for brix %).

sugar + yeast = alcohol, Co2 and heat.

Fermentation is an exothermic process. The internal temperature of the fermentor can be as much as 10F above ambient conditions on the outside, just due to yeast activity.

70 minute boil.

Malt:

- 24 lbs of American 2-row (use local grain from your state/region if you can, support your local farmers)
- 1.2 lb aromatic malt
- 1.5lb cane sugar – [in Belgian beers sugar is added to lighten the body of the beer without affecting the

taste, it will also increase the ABV as it should fully convert. **Warning:** Belgian beers are not Budweiser. drink them responsibly and slow…]

Hops:

- 10 minute into the boil (70 mins total), add 2 ounce of Magnum Hops – for Bitter
- at end of boil, add 2 ounce of Styrian Hops – for Aroma

We always add Irish Moss at 15 minute to end of boil

Yeast:

We used a French Saison #3711 // but there are other choices not limited to: Abbey Ale or Wyeast 3787 (Trappist High Gravity) yeast (we recommend you make a starter atleast 24 hours bore brew day).

Fermentation:

Add 2 days into the fermentation the Apricot Puree, 5-6 lbs. Fruit doesn't transfer well in boil, otherwise skip if you don't want the Apricots.

Arduino + Raspberry Pi to measure fermentation temperature

This is the original Version #1 // it will get you going – please see Version #2 for the dynamic sensor processing and more on plotting the data using an API and code.

temperature logging sensor jig above in fermentor…

Above on the left (Raspberry Pi), middle (breadboard), right (Arduino UNO) – this web site runs on the Pi and you are reading this article right now from it :-)

Home Brewing is more than just the act of beer brewing at home to many people // it is a hobby filled with lots of creativity, ideas and passion. One of our goals was to capture accurate temperature measurements of the fermentation – once you capture the data you can do things with it.

We decided to use a digital 3 wire temperature sensor also referred to as a 1-wire system, because the data is sent over 1 wire, the other two are the volt and ground cable.

We used a DS18S20 Dallas 1-Wire digital thermometer, this sensor is digital and fairly accurate and the program can delivery the data in C or F or whatever you can program for, and it can send the signal over longer distances than an analog thermister. Also you can have multiple digital readers on the same wire, since each one is identified with a digital ID and you can separate the sensors within the programming code.

http://pdfserv.maximintegrated.com/en/ds/DS18S20.pdf

STEP #1

Setup the Arduino + Raspberry driver software // **Google this and do it on your own…**

So in our setup we used an Arduino UNO connected via the USB cable to a Raspberry Pi B, and the Pi also powers the Arduino, get a better 2.0+ Amp power supply for the Pi, ours is 2.5 Amp. You also have to install drivers that allow the two to communicate over the USB cable (serial) connection of the cable.

STEP #1.1

You need to connect the sensor correctly to a 4.7K ohm pull-up resistor, we used a breadboard to help us with the connections, but you can prototype it better. The breadboard will connect to the Arduino, below a simple way to show the connections of the cables. The C code is setup to receive the input on pin 2.

STEP #2

Get a program working in C (language) on the Arduino loaded correctly through the IDE, to read the temperature from the sensor – you will have to learn how to do this part and be overall familiar with the basics of how to use the Arduino.
If you never done this before, (go learn that and then come back to do this step), we are sharing the code that we use below, it compiles fine, you might have to install some dependencies, like the OneWire and Time libraries (learn that too).
[crayon-5b4ccdfe5cc47427323862/]

STEP #3

Use a Python script on the Rasberrpy Pi to read the signal data being sent by the Arduino, see examples of what we actually use right now below. This script not only reads the data from the Arduino over the (serial USB) cable but also logs the data to a (tab delimited) text file while appending date/time stamp for each point reading, it does this every 5 seconds. 5 seconds could be an overkill for your project, maybe you want it every 30 seconds, it all depends what you

are after and the resolution of the data capture that you need.
[crayon-5b4ccdfe5cc52990276628/]
Version #2 of this code is available – http://kodiakbrewing.com/wordpress/?p=4172 // it ignores a temp sample if it incoming the same as the one just recorded, saving space if you are recording remotely over long time.

STEP #4

Do a test and record some data for a few days or a few weeks in the background using (nohup), you will have to learn Linux as well. Run the script basically for a few days or weeks and then learn how to graph the data captured in whatever you see fit best way. Once you have the data in a flat-file, you can transfer it over network and open the data with many different programs to create a temperature/date-time graph. You can also use Python to create the graphs as well, etc…

We used a free program (Plot2) on a Mac to open the flat text file to read the data and it would actually automatically plot a graph. Keep in mind that if you record a test sample of say 2 weeks (of stable temperature that don't vary much), you will see mostly a flat line, but during fermentation you will see a spike of a few days and then a slow decline as the fermentation finishes off – but as tests go for (code and the sensor) – this is a good start.

So think about it, here you learn about the Arduino, and the Raspberry Pi and Linux, and text files to capture the data and C/Python programming languages, and how to graph the data, this is just scratching the surface. You can take this much further, from displaying the temperature live on an LCD screen, to graphing it live on a LCD screen, to writing more program code and maybe even regulate a heater band over the fermentor to control the fermentation temperate after the yeast finishes its job, to deal with off-flavors for example and many other things, not just temperature.

You also see the min() and max() ranges the yeast temperature was reached during the reaction time of the fermentation to see if you hit the manufacturers recommended temp ranges, just yet another example of the data's value.

Bottom line is that you not only learn new things, but capture useful data that you can analyze on and take action with – to in the end improve and make great beer.

Updates will come later with additional data, all our future beers will come with a fermentation charts of the overall process of the yeast used.

Also check out the – https://wizbrewery.wordpress.com/ Waldy the Wiz, also makes a great project and he shares all of his hard work – his is a little bit more advanced than our example.