**BBot interview questions**

**What is the “official” name for this device?**

“BBot!”

**What does it do?**

The snippet from the Maker Faire page is a good overview:

BBot! An open source, remote controlled, drink serving robot that features great examples of how to: interface hardware to a BeagleBone Black, use an FPGA cape, make a .NET GUI that links to the bot via XBee's and more!

Essentially this robot is a remote controlled “cooler camel.” It talks, sees and displays beer advertisements on a screen. It pulls a cooler on a mini trailer and solicits drinks to people at the pool! Obviously!

The remote control data steam from the Microsoft .NET GUI program is transmitted to a BeagleBone Black where the data is parsed and routed to the various control systems on the robot.

The remote control data transport is over a 900MHz XBee pro link. This allows for very long range control (think miles). The video stream is limited to about 300 meters unfortunately – limited by the power of the analog video transmitter – but I’m currently working on running the video and control data link through the cell network instead.

Throughout the code, the names of the classes and variables are trying to be anthropomorphic which is entertaining. For example, rather than calling the code that runs the robot’s wheels “Motors.cpp” or “Drives.cpp” I named them things like “Voice.cpp” and “Legs.cpp.” Functions might be “Walk( int speed );” instead of “Drive ( int speed );.”

Some other fun features include:

1. BeagleBone Black working systems:
   1. SPI, UART, I2C and Digital IO driver setup in one Device Tree Overlay file.
   2. C++ runtime that launches at BBB boot.
      1. Uses Posix multi-threading.
      2. Demonstrates C++ concurrency primitives such as “semaphore.”
      3. Few OOP principles however very good data encapsulation.
   3. Device Tree Overlay, IO setup and system initialization happens on launch of main embedded C++ control program at boot.
   4. C++ code to send and receive data on the SPI bus, serial TTY01 device, I2C interface and digital IO.
   5. Has a nice and simple way to do “register mirroring” to an external device (in this case an FPGA cape) via SPI.
   6. A QT GUI program for displaying advertisements and drink prices on a mini HDMI screen. The QT program can even overlay a debug console which is very helpful.
2. Microsoft .NET GUI program written in C#.
   1. Uses a WPF, MVVM design pattern.
   2. Lots of user interface data binding.
   3. Essentially the main window is filled with user controls that follow the anthropomorphic concept. Controls are named things like Eye Ball, Legs and Voice for example. Each control communicates with one of BBot’s systems.
   4. Use of WPF “Styles.”
   5. Data is passed to the computer’s COM port using .NET standard libraries.
   6. The GUI can connect to a Leap Motion controller (<https://www.leapmotion.com/>) and the robot can then be driven simply by waving your hand around above the sensor!
3. There’s an FPGA cape stacked on top of the BeagleBone Black that provides a robust means of low level hardware interfacing. Very useful in robotics applications.
   1. Created by ValentFx (<http://valentfx.com>). This cape can actually mate with the BBB, the Raspberry Pi and the standard Arduino header! It’s 3 in 1!
   2. Does digital signal generation for servos.
   3. High speed encoder quadrature counter for closed loop motion control.
   4. UART multiplexing for expanded system UART channels.
   5. OpenCores SPI Slave IP core as primary data link to BeagleBone Black’s embedded software.
   6. A well commented, modular Verilog HDL design.
4. System has speakers and a 10 watt audio amp.
5. Has a Real Time Clock chip.
6. Has a text to speech board that works very well. It sounds just like Stephen Hawking!
7. Uses precision 2048 line count encoders for closed loop wheel control.
8. Runs on cheap rechargeable 12v lead acid batteries.
9. Cooler cart has space for more batteries if needed.
10. When people try to steal drinks I sick the robot’s absurdly loud alarm system on them.

**Where did the idea/inspiration for the device come from?**

One day my smart-y-pants friend and I were relaxing at the pool and we ran out of drinks. He leaned over and made some sort of sarcastic comment like “hey man, why can’t you make a robot go get more drinks for us? I can’t bring myself to get up!” I didn’t have a good answer for him… I mean, why not have a robot go get the drinks anyway? It’s 2013 people!

That and I had a BeagleBone classic at the time I was dying to figure out an excuse to work with. I wanted to learn more about Linux and C++ more than anything. It took me a while to realize it but the drink retrieval challenge was the most perfect, not too hard, not too easy, mostly impractical and hilarious idea I could come up with.

**What components do you use in the device?**

I was focused on making the thing as cheap as possible (the robot was completely self-funded…) In order to keep costs low I went with a simple mechanical design. The main chasse consists of some simple disks of Plexiglas stacked on top of each other with threaded steel rods, nuts and washers. Everything could be bought at a hardware store for the mechanical assembly.

Electrical components were bought slowly over a six month period. I used standard industrial “DIN rail components” for the systems power train, voltage regulation and circuit protection. This sort of stuff can be found for cheap at Digikey.com.

The BeagleBone Black was the brain of the device and I was able to prototype most of the robots systems with just a breadboard on my workbench.

The XBee radios were a treat to finally work with. They are so cool and simple to use. At first I bought 2.4GHz model but this conflicted with the 2.4GHz 500mW analog video transmitter I was using. (When I turn this video transmitter on it knocks out people’s wifi hahaha.) When I moved to 900MHz radios it worked perfectly.

Believe it or not Amazon has some good deals on cheap RC hardware for things like hobby airplanes. I got the CMOS camera and the miniature, servo powered pan/tilt camera gimbal on there!

Motors were bought on Pololu.com. I got motors with a 50 to 1 gear ratio because I knew I would need some serious torque to pull a cooler well. Loose calculations said the gear ratio would be high enough to pull a 30 to 40lb load.

Gears, axil and bearings were carefully purchased on McMaster.com and Sdp-Si.com. This actually took a long time to get right and there were more than one erroneous orders placed… I’d be much more focused and thorough if I had to do it again! Turns out motion systems can be a real trick to get right the first time when building from raw components.

The BBot Bill of Materials can be found on the project’s GitHub site.

**Why did you choose BeagleBone Black for this device instead of a different board?**

There are many reasons. First is cost. There’s nothing like it with as much computing power, flexibility, and size of active community at the $45 price point.

I knew the Raspberry Pi is very popular however it seemed to be geared more toward video applications. I needed the vast number of IO found on the BeagleBone Black for my robot. I love how small it is too!

Although I haven’t had time to try it out I understand that the Black’s programmable real-time unit (PRU) can be setup to count a high speed encoder quadrature signal or even do digital signal generation (instead of using PWM channels). This would eliminate the need of and FPGA cape some day! Indeed, BBB’s PRU features open the doors to some very interesting and fun possibilities.

Another reason for going with the BBB is I knew that at some point I would like to test out the Yocto project on the BeagleBone Black. The Yocto project (<http://www.yoctoproject.org>) supposedly supports the BeagleBone Black which is super sweet.

At the project outset I had hoped to make the BBot do some sort of autonomous, machine-vision based application as well. For this you need some serious computer power which is way outside the reach of most microcontroller platforms like the Arduino series or a PIC.

Finally, having a (Linux) operating system working for you allows for robust and complex software applications that would otherwise be much, much more difficult to build.

**What challenges did you run into while creating the device, and how did you overcome them?**

I studies electrical engineering in school so the electrical aspect of the robot wasn’t too bad. Most of the time I would have problems with the software development side of the project. Getting drivers working on the BBB took some serious trial and error effort for example.

Although the BBB is rather new, there are an amazing number of sources on the internet that show you how to do the common things (ie. setting a static IP address via connman, setting up your cross compiling environment or launching programs on boot).

There’s a professor out of Ireland named Derek Molloy who has a killer video series on YouTube (<http://www.youtube.com/user/DerekMolloyDCU>) about getting started with the BeagleBone too. I’d say I picked up the most information from him. He brings you from zero to hero in no time! It’s great.

Fortunately, when I would run into really hard problems relating to things like system architecture or C++ details I was able to ask some the seasoned software professionals I work with. Knowing experts in the computer / software engineering field is very helpful at times!

**Where can we send people for more information on the device?**

Working source code, photos, video and documentation can be found at:

<https://github.com/andygikling/BBot>

This is the project’s page for now. Documentation is a bit thin for now. See the block diagram to get a better feel for the system topology.

<http://valentfx.com> is where you can find information about the FPGA cape I’m using. It’s not for sale yet but stay tuned. We’re trying to get a Kickstarter going or something. This board is real slick, extremely powerful and easy to use. You can drop your compiled binary on it with nothing more than a USB cable!

**What will your next project be?**

I think I’m going to keep adding features to this platform. I’ve spent enough money on the drivetrain and electronics so it doesn’t make sense for me to start over on anything new right now.

I’d like to potentially add a precision pan / tlit camera system and incorporate this tracking algorithm called TLD (<http://personal.ee.surrey.ac.uk/Personal/Z.Kalal/tld.html>). “Annoying follower bot” would be pretty cool / funny.

I’ll be adding GPS probably and improving the Leap Motion performance.

I’d like the main data, video and audio link to run through the cell network so I can approach people on the street for comedic purposes. I think that would be pretty funny… and who can say no to a free Coke from a friendly robot that seems to have no driver?

**Are there any other interesting facts you want us to know about you?**

I work for LasX Industries Inc. in Minnesota. We make the best darn industrial laser systems money can buy!

…and I’m 14 feet tall. Seriously, 14 feet tall - it’s a problem.

;)