Raspberry Pi Robotics

CyberData Corporation
Innovation For a Global Future

Lloyd Moore, President

Lloyd@CyberData-Robotics.com

www.CyberData-Robotics.com

Seattle Robotics Society 6/21/2014

Outline

Why the Pi?

Why not the Pi?

Raspberry Pi – PSoC Development Board

RPi-Dev Motor Controller

RPi-Dev SPI Ring

System Block Diagram

Resources

Why the Pi?

There are TONS of other controller boards on the market, what makes the Pi special?

- Computing power per dollar is, or very close to best in class
  - 700MHz ARM 11, hardware floating point, Open GL GPU, 512MB RAM, Ethernet
  - USB, SD Card for mass storage, audio output, video output
  - $25 - 35 price tag!
- 5 MP camera available for $25
  - Plugs directly into the Pi and software is available for vision applications
- I/O Expansion port
  - Easy to access
  - Contains UART, I2C, SPI and several GPIO lines
- Full schematics available for the board
- Fairly light weight and power efficient (1.6 oz, 3.5W)

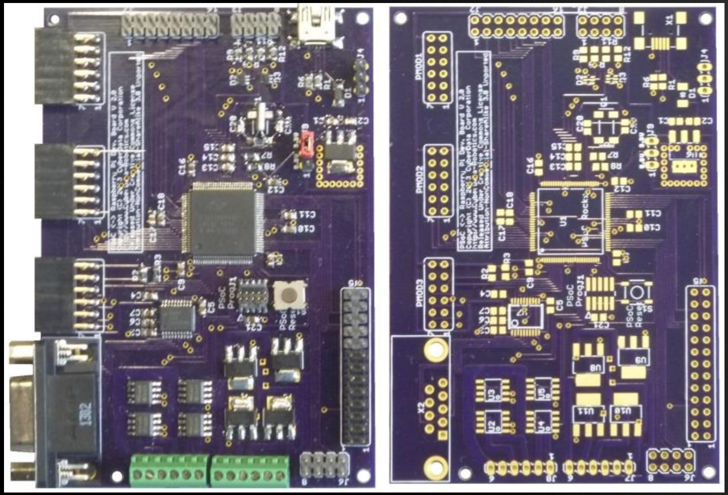
Why not the Pi?

But nothing is perfect right?

- No real time processor – could emulate with Linux, but not great
- No analog inputs, no ADC
- Minimal GPIO and what you do have is 3.3V and raw, don't even have a proper RS-232 port
  - GPIO also awkward to get at from inside Linux
- No motor drivers, or high current outputs
- Really no standard I/O of any kind other than USB and Ethernet!
- Doesn't really have enough processing power to do vision applications

Raspberry Pi <-> PSoC Dev. Board

- Designed to be a flexible development and prototyping platform for both Raspberry Pi and PSoC ecosystems
- Developed this as I frequently need to rapidly prototype ideas and wasn't finding anything single device available that really fit the bill
- CyberData (my company) is releasing this as open source!
    - Designs and documentation released under Creative Commons, Attribution-NonCommercial 3.0 Unported license
    - Sample code released under GNU GPL V3.0 license
    - If someone really wants a commercial license talk to me about it

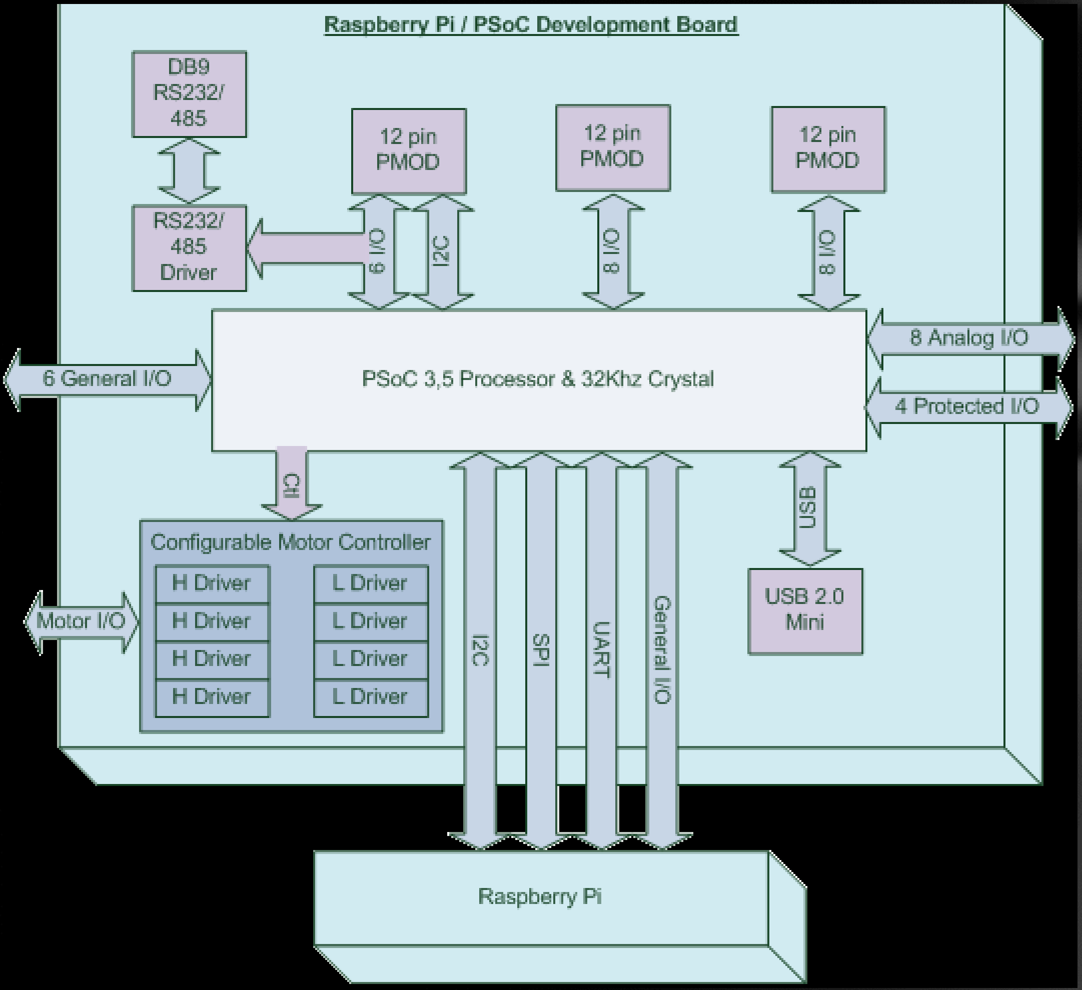


RPi-Dev Features

Provides the Raspberry Pi with the resources for doing real word, real time problems:

- Central processor is a PSoC 5LP or PSoC 3, 100 pin device
  - Allows for real time work on the Cortex M3 core and configurable logic in the UDB blocks for “hard real time” applications
  - Has configurable analog and DSP processor for low level signal processing
- Standard RS-232/485 driver on board, with a DB9 connector
- 3x 12 pin Diligent PMOD connectors – many devices available
- Configurable motor controller
  - Directly drive 2 3.5A DC motors
  - Can also be used for stepper motors, solenoids or other high current loads
- Additional GPIO lines, 8 precision analog, 4 protected, 6 general
- Also usable on any PC with a USB connection, or standalone
  - Board can be powered from the USB connection as well

RPi-Dev Block Diagram

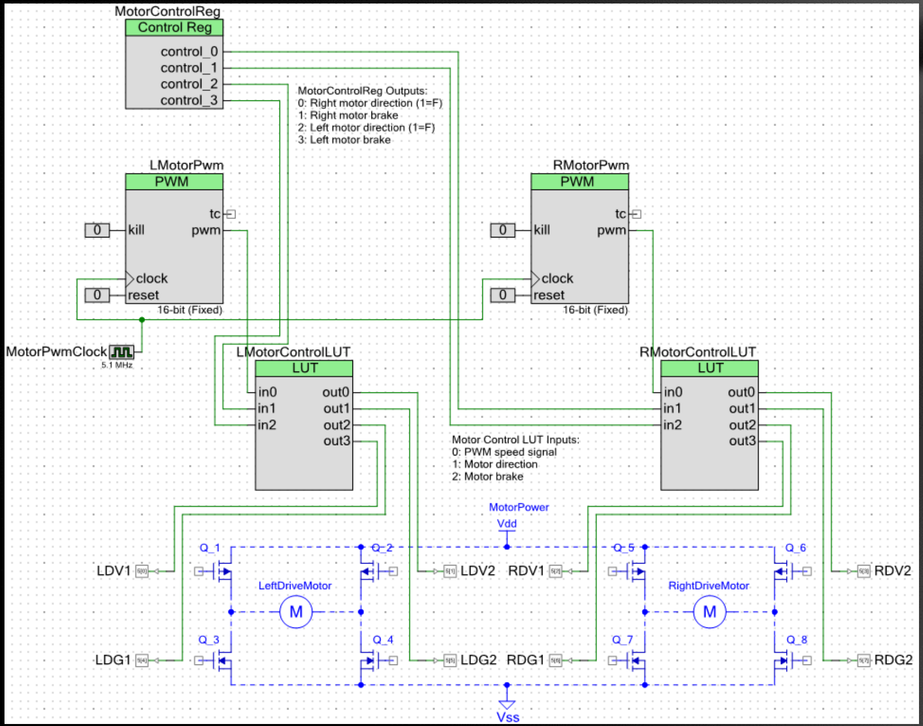


RPi-Dev Motor Controller

Designed to provide a very flexible resource which can be used for most types of motors

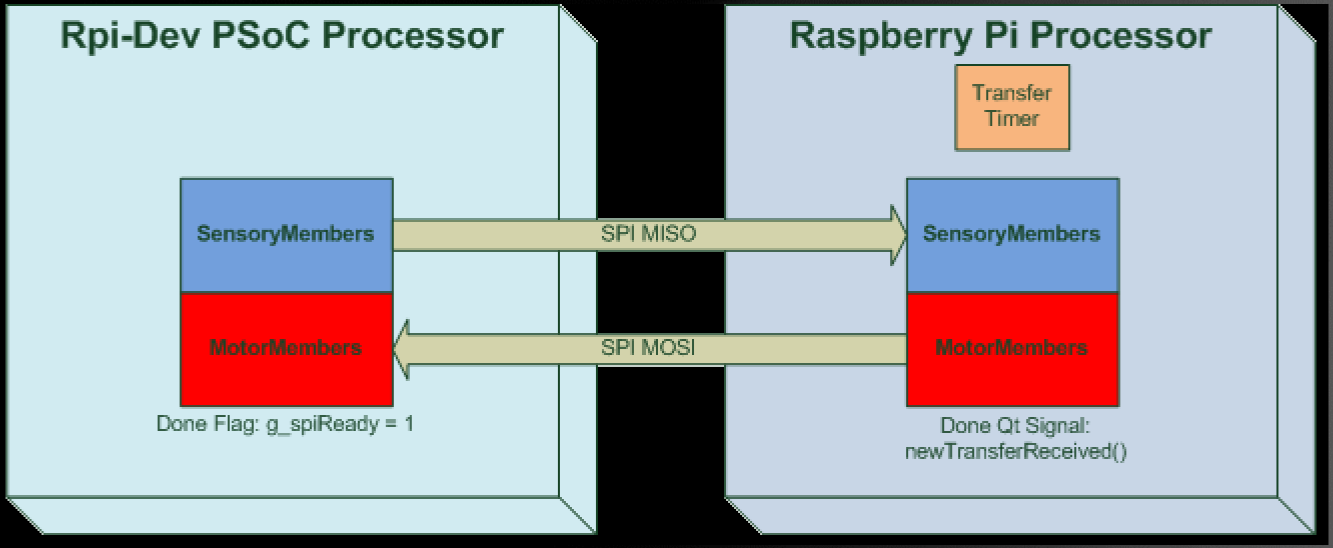
Achieved by having robust discrete motor drivers on the board, while keeping the control logic configurable in the PSoC UDB system

Sample code implements a dual H-Bridge based motor controller with PWM speed control



RPi-Dev SPI Ring

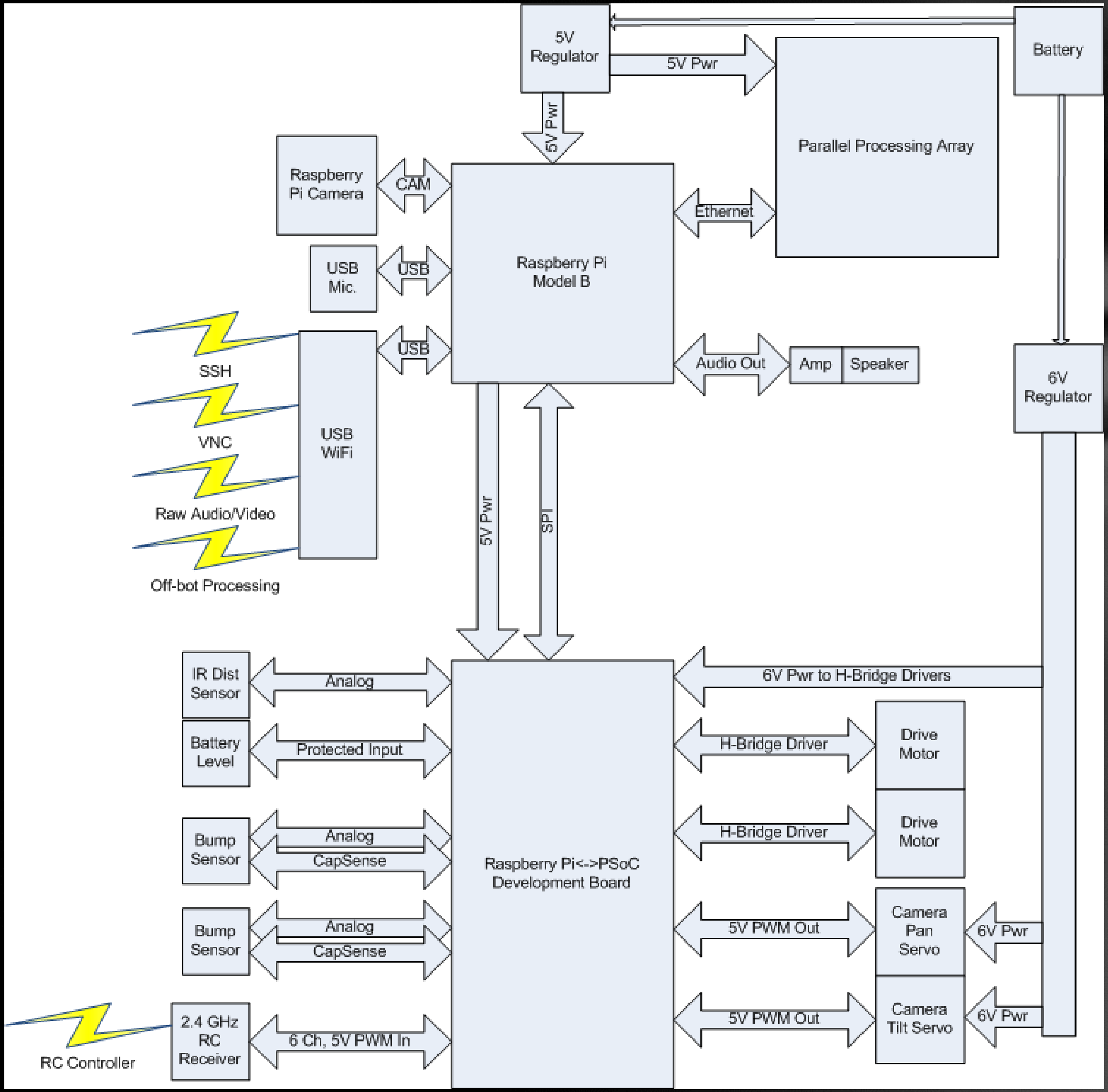
- Provides generalized, high speed, packet based communications between the Raspberry Pi and the PSoC processor
  - Uses the SPI Master peripheral on the Raspberry Pi
  - Uses a SPI Slave component with DMA support on the PSoC
  - Transfer made at 4Mbps, bidirectional
  - Packets appear in a C structucture on each side at a configurable time interval
- All this is implemented in the sample code – just configure the data structure and hook up the completion signals to your own code!



To use the SPI Ring

1. Define the *SensoryMembers* and *MotorMembers* data structures to contain the data members you want in the file *ringcoms.h*
2. Note that both structures MUST be the same size, add padding if not. If the structures are not the same size you will get an error!
3. In the PSoC main() routine change the code testing the g_spiReady flag to do what you need to at the end of each transfer.
    1. For a robot this will be transferring the data in the MotorMembers structure to the actual motors and reading new sensor data into the SensoryMembers structure.
    2. Note that you can also hook the DMA interrupt directly if desired.
4. In the BotMonitor program connect the newTransferReceived() signal to your handler function and do what is needed there.
    1. For a robot this will be processing the sensor data and issuing a new set of motor commands.

SRS Robot Block Diagram

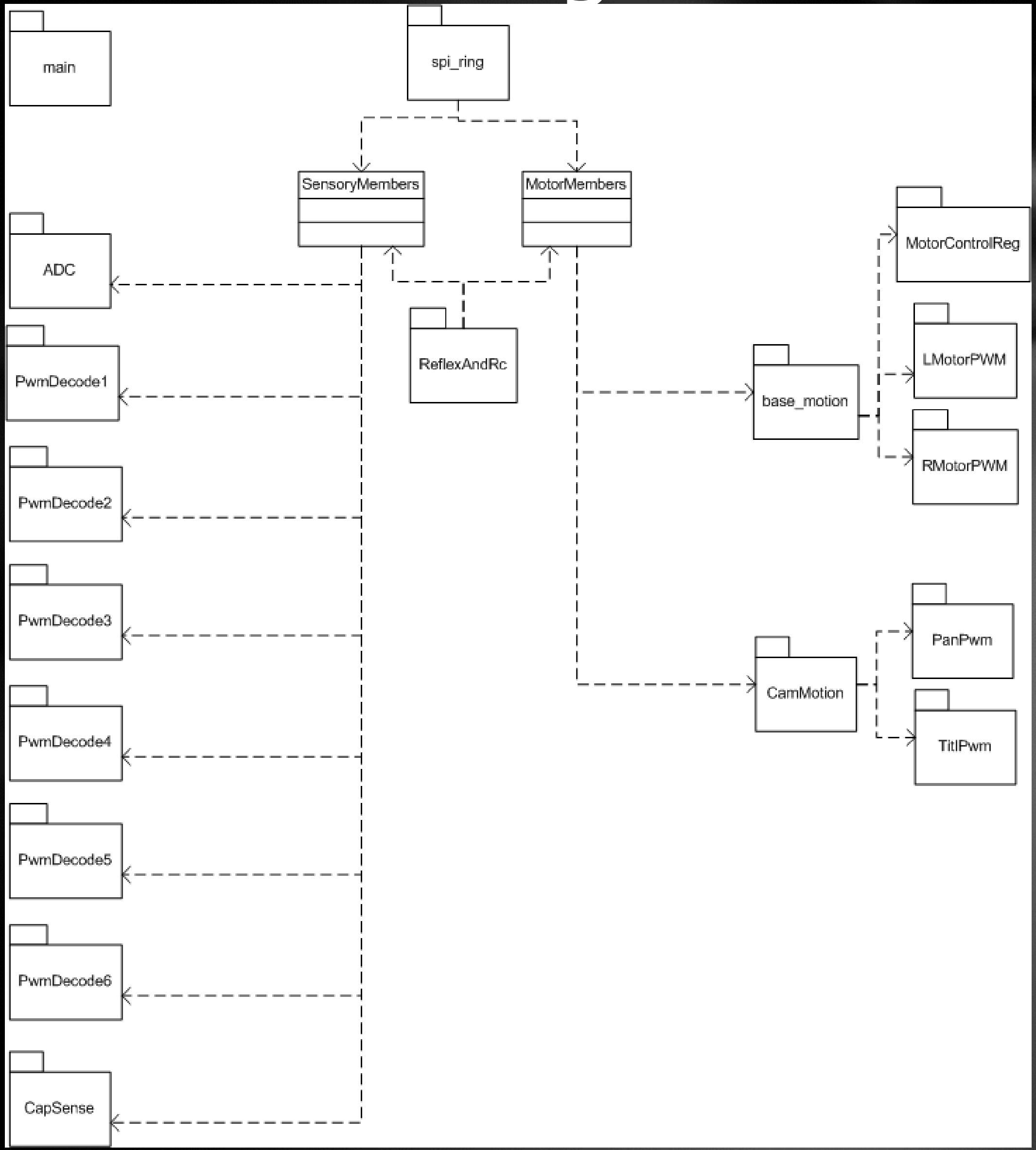


SRS Robot PSoC Software Diagram

Each new SPI packet:

- If bump sensor triggered, block forward motion
- If RC enabled, pass RC commands to motors
- If RC not enabled pass MotorMembers commands to motors
- Read sensors and populate SensoryMembers data structure
- Wait for next SPI packet

Note: This algorithm is NOT contained in the sample code.



References

Raspberry Pi <-> PSoC Board Open Source Resources: http://www.cyberdata-robotics.com/RPi-Dev

Raspberry Pi Home Page: http://www.raspberrypi.org/

Tight VNC Home Page: http://www.tightvnc.com/

Qt Home Page: http://qt-project.org/

PSOC Home Page: http://www.cypress.com/psoc5lp/?source=CY-ENG-HEADER

QExtSerialPort Home Page: http://code.google.com/p/qextserialport/

BCM2835 Drivers: http://www.open.com.au/mikem/bcm2835/index.html

BCM2835 Peripherals Datasheet: http://www.raspberrypi.org/wp-content/uploads/2012/02/BCM2835-ARM-Peripherals.pdf

MagPi Magazine: http://www.themagpi.com/

Questions?