Team 832 - Chimera

2009 FRC Control System Workshop
November 15, 2008

Schedule

- **Schedule (rough)**
  - 8:30 am Check-in
  - 9:00 am LabVIEW/System Overview
  - 10:30 am C/C++
  - 11:30 am Hardware review
  - 12:00 pm Questions
  - 1:00 pm Teams Leave

Presenters

- Jeremy Roberts……. C/C++
- Brett Dutro ……. Vision System
- Rick Folea ……. Labview
- Supporting Cast:

RoboJackets

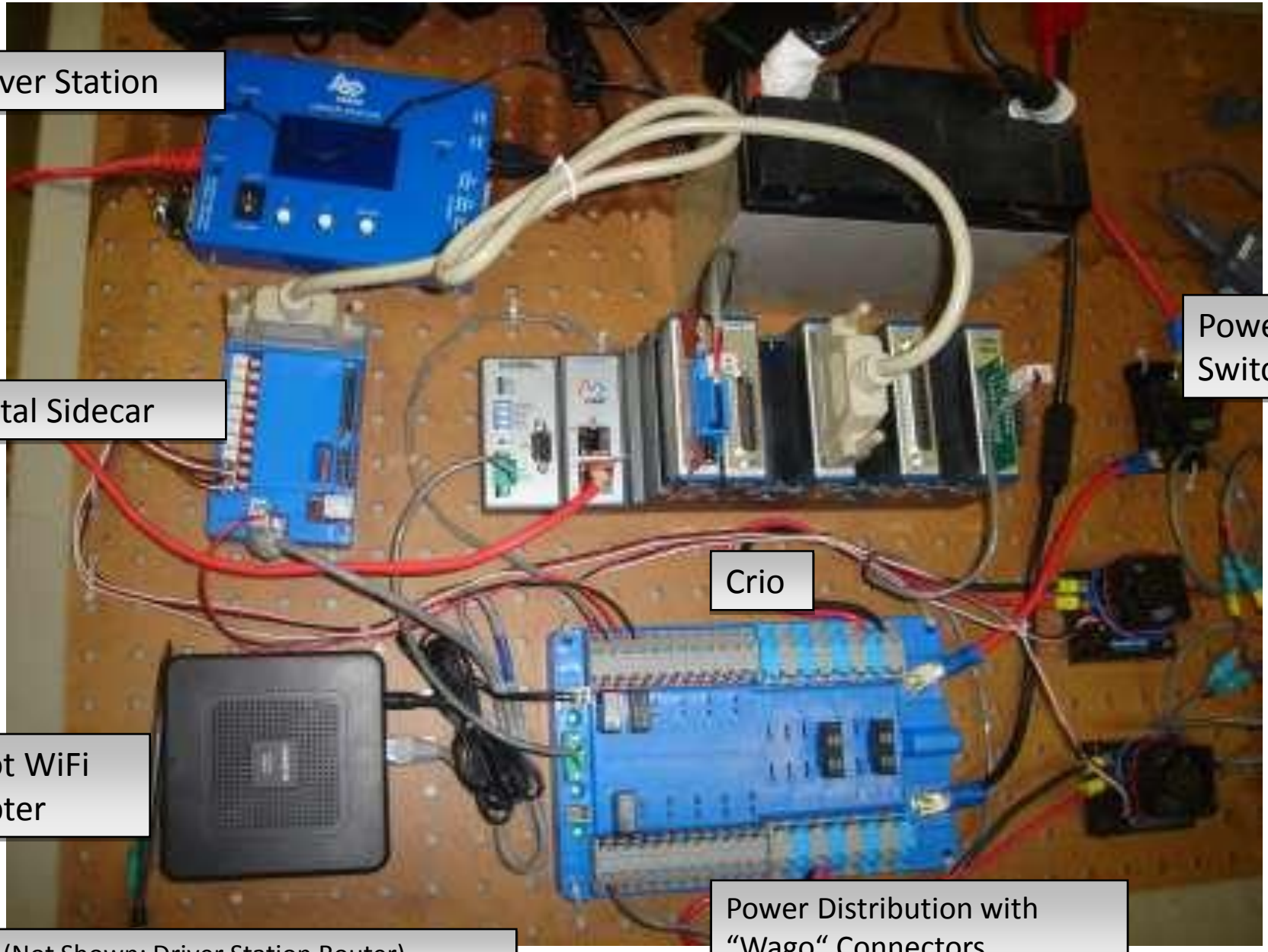
- Elan Grossman
- Philip Marks
- Stefan Posey
- Roman Shtylman
- Andy Bardigy

Forsyth Alliance

- Dana Burnham
- Will Booker
- Drew Martin
- Chris Folea
- Tyler Burnham

System Overview

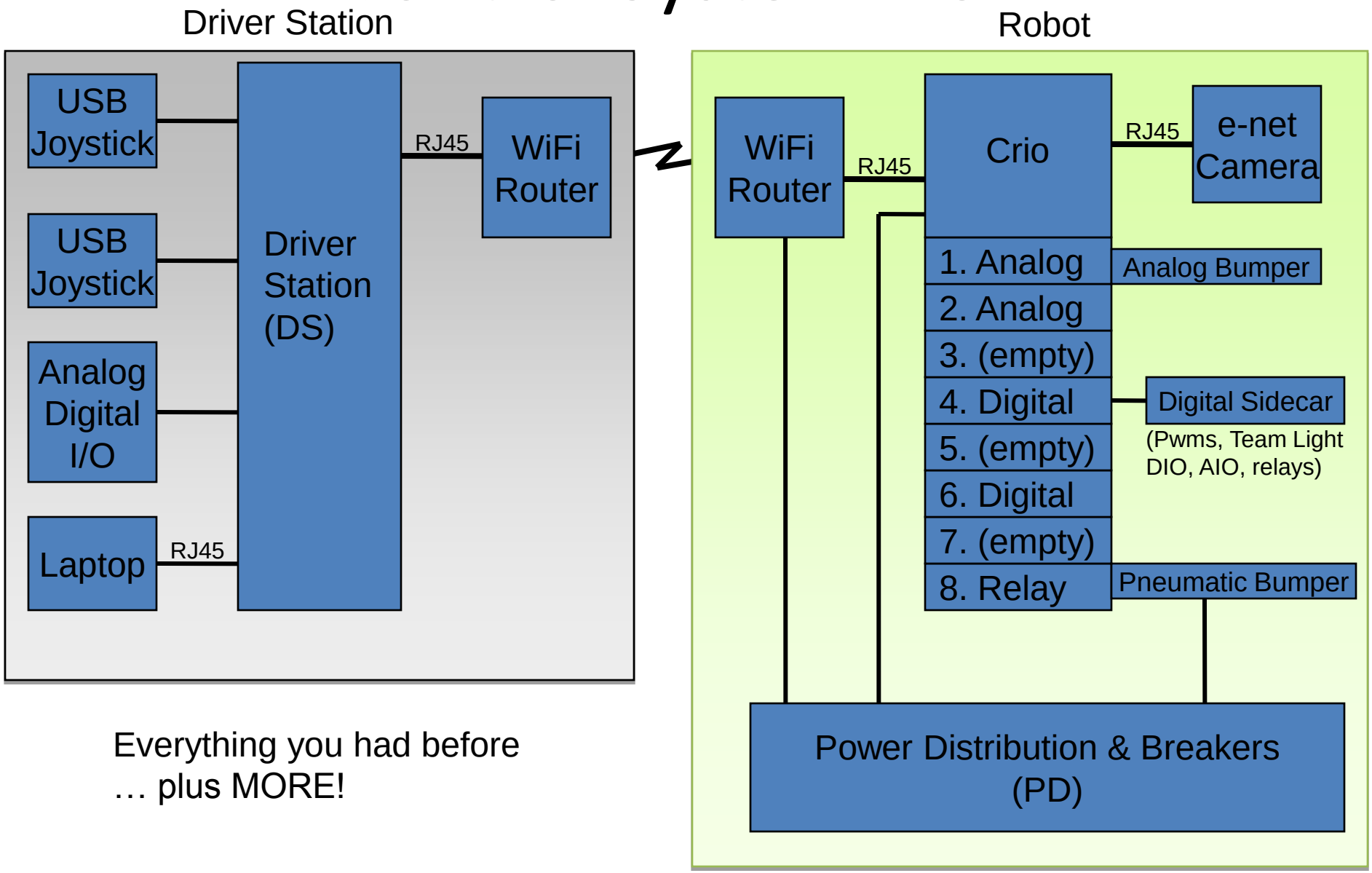
Control System



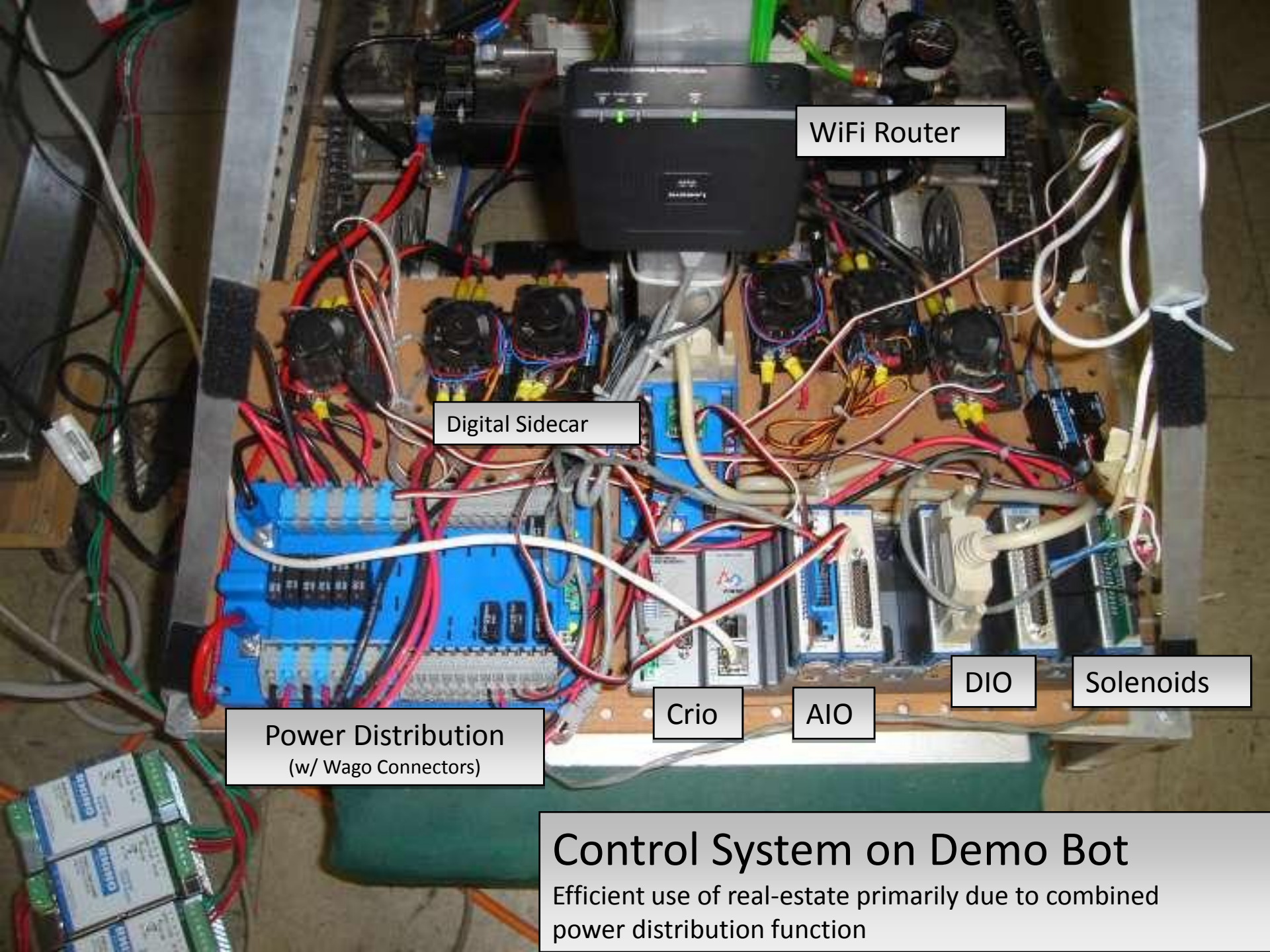
Control System

(Different Angle)

Control System Flow



Everything you had before
… plus MORE!



Control System on Demo Bot

Efficient use of real-estate primarily due to combined power distribution function

Features Comparison

	2008	2009 KOP	2009 Alt*
DIO	18	14	28
AIO	16	8	16
Relay	8 (fwd + rev)	8 (fwd + rev)	16
PWM	16	10	20
I2C	---	1	2
SPI	---	1	2
Solenoid	---	8	8
Comm	RF Modem	Wifi	Wifi
Power	Kludge	Clean	Clean

***2009 alt – If you add additional Sidecar and bumpers**

New Features

- The following are now done in Hardware
- Counters
- Accumulators (for Gyro or DAA)
- Encoders (1x, 2x, 4x)

Other new features

- I2C
- SPI
- Ethernet camera – a “real” video camera
- WiFi control
- Solenoids – no need for Spikes on valve control!

LabVIEW Demonstration

Introduction to C++

Thanks to team 1114 for a few of their examples to make my life easier when putting this together last night.

Outline

- Configuring the cRio for C++ development
- Development environment/workspace
- Creating a new default FRC project
- High level architecture of the default code
- Downloading to the robot
- WPI Library (a few key classes and how to use them)
- Debugging using printf
- Vision

Development Environment

- Created by WindRiver (embedded system)
- Workbench IDE based on Eclipse
- Compiler used is GCC
- VxWorks Real-Time Operating System (RTOS)
- Integrated program downloader

2009 Default Code Architecture

- Three main parts to “SimpleRobot”
  - Constructor
    - Called once when the robot is turned on or reset
    - Initialization of objects/variables can be done here
  - Autonomous
    - This gets executed when the system is in auto mode. After execution the cRio must be reset in order to enter auto mode again.
  - OperatorControl
    - This gets executed any time the system in in tele-op mode.

WPI Library

- A few important classes
  - Joystick
  - RobotDrive
  - SpeedController
  - Relay
  - Compressor
  - Solenoid
  - Gyro
  - Encoder
  - AnalogChannel

Joystick

Represents 1 of the 4 USB Drivers Station inputs

Constructor:

Joystick(UINT32 port)

Important functions:

float GetX(JoystickHand hand = kRightHand);
float GetY(JoystickHand hand = kRightHand);
bool GetTrigger(JoystickHand hand = kRightHand);
bool GetRawButton(UINT32 button);

Example:

```
Joystick *stickL;
stickL = new Joystick(1);              //USB Port 1

float yaxis = stickL->GetY();
If(stickL->GetTrigger()){ /* Do Something */}
```

RobotDrive

Built in class for controlling a robot drive

Constructor:

RobotDrive(UINT32 leftMotorChannel, UINT32 rightMotorChannel, float sensitivity = 0.5)

Important functions:

void Drive(float speed, float curve);

void TankDrive(GenericHID *leftStick, GenericHID *rightStick);

void ArcadeDrive(GenericHID *stick);

Example:

```
RobotDrive *myRobot;
myRobot = new RobotDrive(1, 2);

myRobot->ArcadeDrive(stickL);
```

SpeedController

Either the Victors or Jaguars connected to a PWM output on the digital sidecar (Output from -1.0 to 1.0)

Constructor:

Victor(UINT32 channel)

Jaguar(UINT32 channel)

Important functions:

void Set(float value);

Example:

```
Victor *arm;
arm = new Victor(1);   // PWM 1
arm>Set(1.0);          // Full speed
Wait(1000);
arm>Set(-1.0);         // Reverse direction full speed
```

Relay

Used to control a relay plugged into one of the relay outputs on the digital sidecar

Constructor:

Relay(UINT32 channel)

Important functions:

void Set(Value value); //typedef enum {kOff, kOn, kForward, kReverse} Value;

Example:

```
Relay *spinningLight;
spinningLight = new Relay(2);       // Relay 2

spinningLight->Set(Relay::kForward);
```

Compressor

Used for starting the compressor with associated digital input for pressure switch

Constructor:

Compressor(UINT32 pressureSwitchChannel, UINT32 compressorRelayChannel)

Important functions:

void Start(void);

UINT32 GetPressureSwitchValue(void);

Example:

```
Compressor *compressor;
compressor = new Compressor(1, 1); // Pressure switch on relay 1 and digital I/O 1

Compressor->Start();
```

Solenoid

Solenoid value plugged into the pneumatics module on the cRio. (Use 2 for a double solenoid)

Constructor:

Solenoid(UINT32 channel)

Important functions:

void Set(bool on);

Example:

```
Solenoid *sNoid[2];
for(int i = 0; i < 2; i++)
{
	sNoid[i] = new Solenoid(i + 1);
}

sNoid[0]->Set(true);    //Sets solenoid output 1 to true
```

Gyro

Used for an angular rate sensor connected to analog input 1. Input 1 has an accumulator built into the FPGA.

Constructor:

Gyro(UINT32 channel)

Important functions:

float GetAngle(void);

void SetSensitivity(float voltsPerDegreePerSecond);

void Reset(void);

Example:

```
Gyro *gyro;
gyro = new Gyro(1);                  // Analog input 1

gyro->SetSensitivity(0.0122);        // Set sensitivity for a particular model of gyro
float robotAngle = gyro->GetAngle();
```

Encoder

Used for a quadrature encoder. Your output is x 4 since both rising and falling edges are counted (2 channels).

Constructor:

Encoder(UINT32 aChannel, UINT32 bChannel, bool reverseDirection = false)

Important functions:

void Start(void);

INT32 Get(void);

void Reset(void);

bool GetDirection(void);

Example:

```
Encoder *leftEncoder;
leftEncoder = new Encoder(12, 11);          // A channel is DI/O 12, B  channel is 11

leftEncoder->Start();
INT32 distance = leftEncoder->Get();
```

AnalogChannel

Used for analog sensors plugged into the cRio analog module (12-bit).

Constructor:

AnalogChannel(UINT32 channel)

Important functions:

INT16 GetValue(void);

float GetVoltage(void);

Example:

```
AnalogChannel *armPot;
armPot = new AnalogChannel(6);              // Analog input 6

INT16 armValue = armPot->GetValue();        // Ranges from -2048 to 2047
float armVoltage = armPot->GetVoltage();    // Ranges from -10V to 10V
```

Debugging

- Using printf
  - Used to display text and variables on the Workspace terminal window

Examples:

```
printf("If I get here this code does not work\n");

int luckyNumber = 7;
printf("My lucky number is %d", luckyNumber);
```

C++ Vision API

About the AXIS 206 Camera

- Network camera
  - Accessed through TCP/IP over Ethernet
    - Need crossover cable to connect to cRIO
  - Built-in HTTP and FTP servers
    - Runs Linux
- Uses VAPIX API
  - Exposure and White Balance
    - Auto Mode
    - Presets
    - “Hold current” - saves settings, even after a power cycle
    - Currently no manual settings option

StartCameraTask

int StartCameraTask(int frames, int compression, ImageSize resolution, ImageRotation rotation)

Return value:

- **int** - On success, this function returns the task ID. On failure, this function returns -1. To get extended error information, call GetLastError().

Parameters:

- **frames** – Camera frame rate (1 – 30)
- **compression** – The amount of image compression (0-100)
- **resolution** – Determines number of pixels in the image. Possible values are k640x480, k320x240, k160x120
- **rotation** – If the camera is not mounted in the normal position, the image returned may be rotated to compensate. Possible values for ImageRotation are: ROT_0, ROT_90, ROT_180, ROT_270

GetTrackingData

TrackingThreshold GetTrackingData (FrcHue hue, FrcLight light)

Return value:

- **TrackingThreshold** – This function returns a structure containing HSL ranges.

Parameters:

- **hue** – A predefined hue. Valid values are RED, GREEN, BLUE, YELLOW, ORANGE, PURPLE, WHITE, and PINK.
- **Light** - A predefined type of light. Valid values are PASSIVE_LIGHT, BRIGHT_LIGHT, ACTIVE_LIGHT, WHITE_LIGHT, and FLUORESCENT.

FindColor

```
int FindColor(frcHue color, ParticleAnalysisReport *trackReport)
int FindColor(const Range* hueRange, ParticleAnalysisReport *trackReport)
int FindColor(const Range* hueRange, int minSaturation, ParticleAnalysisReport *trackReport)
```

Return value:

- **Int** - On success, this function returns 1. On failure, this function returns 0. To get extended error information, call GetLastError()..

Parameters:

- **color** – One of several predefined color ranges (see Appendix).
- **hueRange** - The range for hue.
- **minSaturation** - The minimum value for saturation (maximum will be 255).
- **trackReport** – On return, pointer to structure containing values concerning the color area found.

Advanced Processing

- Crop/Scale
- Histogram
- Pixel Value Access
- Particle Filtering
- Morphological Transformation
- Reject Border Particles
- Particle Analysis
- Image Enhancement
- Thresholding
- Send Images to PC

Vision Example

Initialization

```
#include "AxisCamera.h"
#include "BaeUtilities.h"
#include "TrackAPI.h"

#define MIN_PARTICLE_TO_IMAGE_PERCENT 0.25
#define MAX_PARTICLE_TO_IMAGE_PERCENT 10.0

    SimpleTracker(void) {
        if (StartCameraTask(10, 0, k160x120, ROT_0) == -1) {
            dprintf( LOG_ERROR,"Failed to spawn camera task; Error code %s",
                    GetVisionErrorText(GetLastVisionError()) );
        }
        tdata = GetTrackingData(GREEN, FLUORESCENT);
    }
```

Vision Example
Autonomous Code

```
void Autonomous(void) {
    ParticleAnalysisReport par;
    while (IsAutonomous()) {
        Wait(50);
        if (FindColor(IMAQ_HSL, &tdata.hue, &tdata.saturation, &tdata.luminance, &par)
            && par.particleToImagePercent < MAX_PARTICLE_TO_IMAGE_PERCENT
            && par.particleToImagePercent > MIN_PARTICLE_TO_IMAGE_PERCENT) {

                myRobot->Drive((float)1.0, (float)par.center_mass_x_normalized);
        } else {
            myRobot->Drive(0.0, 0.0); // stop robot
        }
    }
}
```

LabVIEW Samples & More at

http://wiki.robojackets.org/w/FIRST_Resources