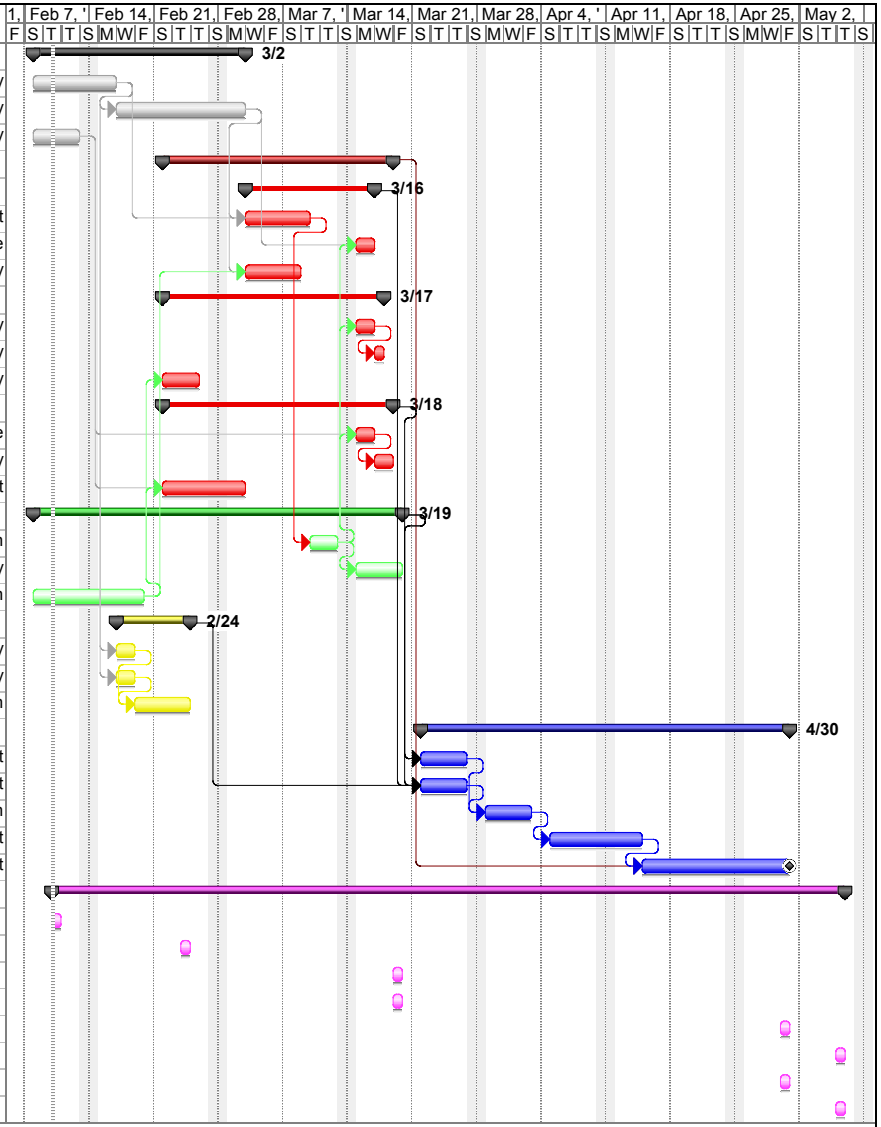
A Appendices

A.1 Appendix: Gantt Chart

ID	Task Name	Duration	Start	Finish	Predeces	Responsible Person	Difficulty
1	**Ordering Parts**	**17 days**	**Mon Feb 8**	**Tue Mar 2**		**Ewout**	
2	Cars Received	7 days	Mon Feb 8	Tue Feb 16			easy
3	Wheel Encoders Received	10 days	Wed Feb 17	Tue Mar 2	2		easy
4	Infrared Cameras Received	5 days	Mon Feb 8	Fri Feb 12			easy
5	**Sensor Component**	**19 days**	**Mon Feb 22**	**Thu Mar 18**			
6	**Wheel Encoders**	**10 days**	**Wed Mar 3**	**Tue Mar 16**		**Ewout**	
7	Physical Mounting	5 days	Wed Mar 3	Tue Mar 9	3,2		difficult
8	Power and Wiring	2 days	Mon Mar 15	Tue Mar 16	19,3		average
9	Communication with Processor	4 days	Wed Mar 3	Mon Mar 8	21,3		easy
10	**Ultrasonic Rangefinders**	**18 days**	**Mon Feb 22**	**Wed Mar 17**		**Ewout**	
11	Physical Mounting	2 days	Mon Mar 15	Tue Mar 16	19		easy
12	Power and Wiring	1 day	Wed Mar 17	Wed Mar 17	11		easy
13	Communication with Processor	4 days	Mon Feb 22	Thu Feb 25	21		easy
14	**Infrared Cameras**	**19 days**	**Mon Feb 22**	**Thu Mar 18**		**Tri**	
15	Physical Mounting	2 days	Mon Mar 15	Tue Mar 16	19,4		average
16	Power and Wiring	2 days	Wed Mar 17	Thu Mar 18	15		easy
17	Communication with Processor	7 days	Mon Feb 22	Tue Mar 2	21,4		difficult
18	**Processing Component**	**30 days**	**Mon Feb 8**	**Fri Mar 19**		**Kyle**	
19	Physical Mounting	3 days	Wed Mar 10	Fri Mar 12	7		medium
20	Power	5 days	Mon Mar 15	Fri Mar 19	19		easy
21	Basic Functionality	10 days	Mon Feb 8	Fri Feb 19			medium
22	**Vehicle Component**	**6 days**	**Wed Feb 17**	**Wed Feb 24**		**Heather**	
23	Electronic Steering Controller	2 days	Wed Feb 17	Thu Feb 18	2		easy
24	Steering Servomechanism	2 days	Wed Feb 17	Thu Feb 18	2		easy
25	Communication with Processor	4 days	Fri Feb 19	Wed Feb 24	23,24		medium
26	**Algorithm Component**	**30 days**	**Mon Mar 22**	**Fri Apr 30**		**All**	
27	Vehicle Tracking	5 days	Mon Mar 22	Fri Mar 26	14,18		difficult
28	Point to Point Navigation	5 days	Mon Mar 22	Fri Mar 26	6,18,22		difficult
29	Vehicle-Following Algorithm	5 days	Mon Mar 29	Fri Apr 2	28,27		medium
30	Path Following Algorithm	8 days	Mon Apr 5	Wed Apr 14	29		difficult
31	Optimization	12 days	Thu Apr 15	Fri Apr 30	30,5		difficult
32	**Deliverables**	**62 days?**	**Wed Feb 10**	**Thu May 6**		**All**	
33	Written Proposal	1 day?	Wed Feb 10	Wed Feb 10			
34	Draft Project Summary	1 day?	Wed Feb 24	Wed Feb 24			
35	PDR Presentations Finished	1 day?	Fri Mar 19	Fri Mar 19			
36	Intermediate Assessment	1 day?	Fri Mar 19	Fri Mar 19			
37	Final Project Summary	1 day?	Fri Apr 30	Fri Apr 30			
38	Website	1 day?	Thu May 6	Thu May 6			
39	Final Project Presentation	1 day?	Fri Apr 30	Fri Apr 30			
40	Final Project Report	1 day?	Thu May 6	Thu May 6			

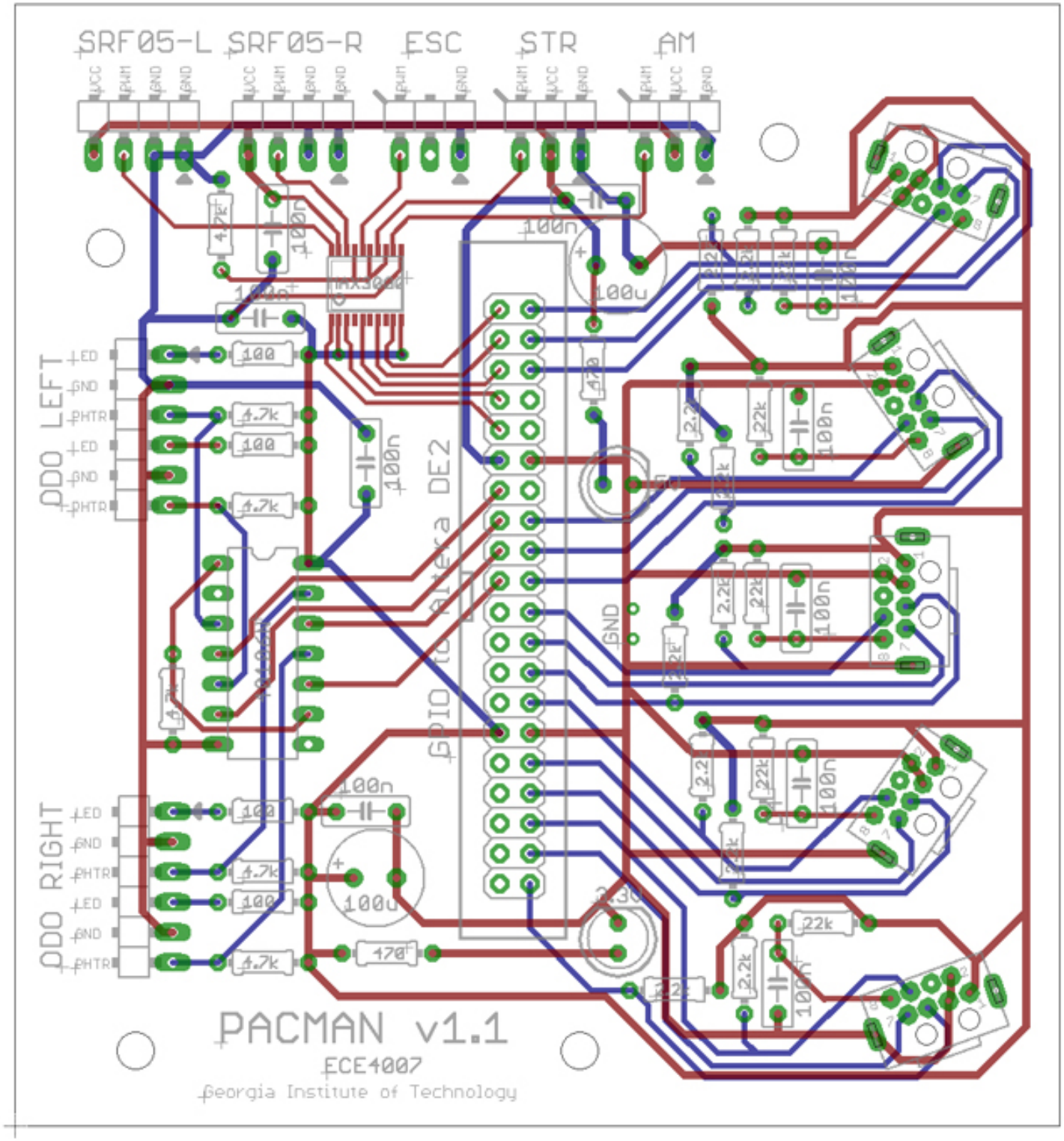


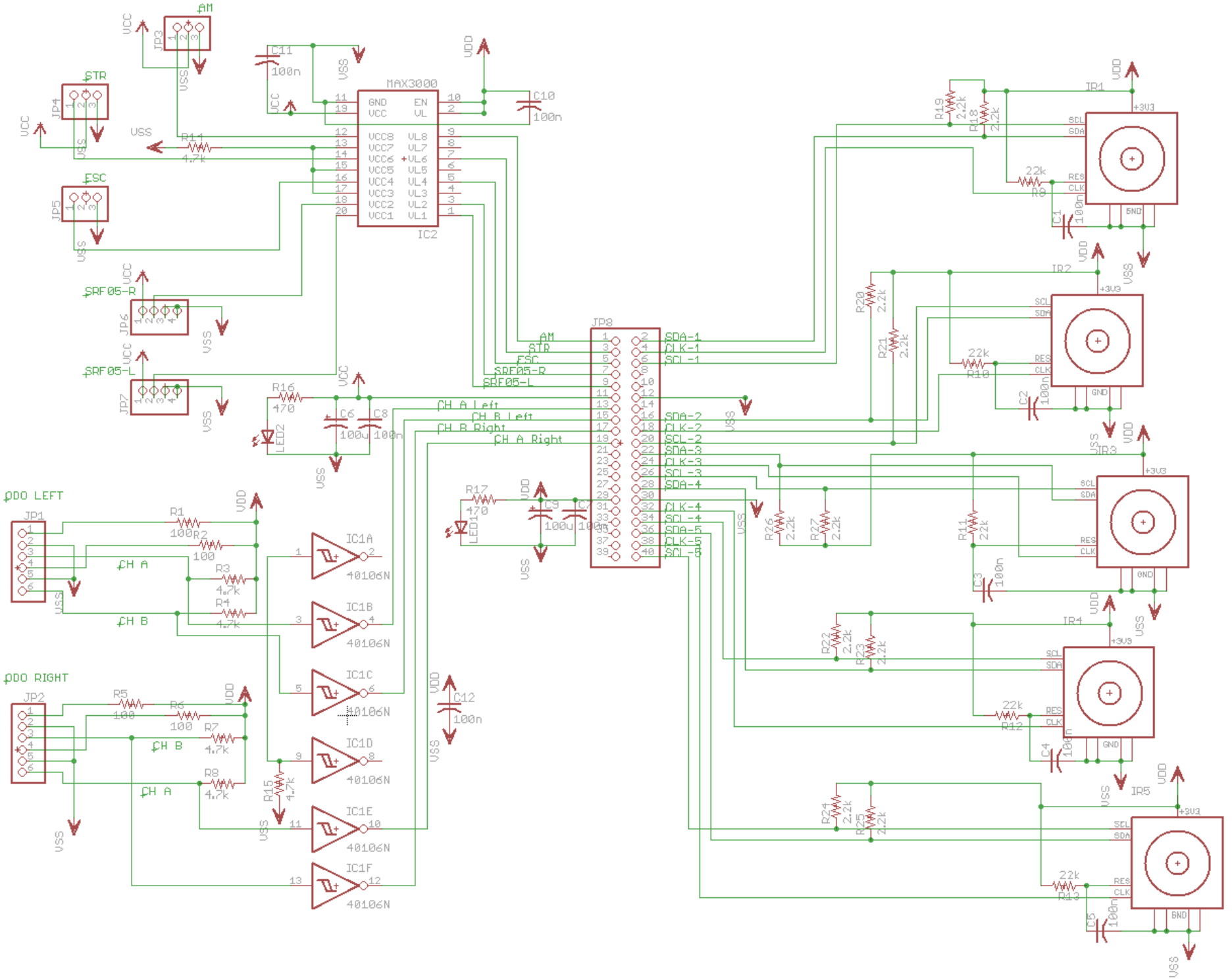
Project: schedule
Date: Wed Feb 10

Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline

A.2 Appendix: I/O Daughter Board Design and Component Connections

I/O Daughter Board EAGLE Layout & Circuit




AM
STR
ESC
SRF05-R
SRF05-L
MAX3000
IC2
JP8
CH A Left
CH B Left
CH B Right
CH A Right
SDA-1
CLK-1
SCL-1
SDA-2
CLK-2
SCL-2
SDA-3
CLK-3
SCL-3
SDA-4
CLK-4
SCL-4
SDA-5
CLK-5
SCL-5
LED2
LED1
IR1
IR2
IR3
IR4
IR5
+3V3
SCL
SDA
RES
CLK
GND
22k
100n
100u
470
2.2k
4.7k
100
JP1
JP2
JP3
JP4
JP5
JP6
JP7
CH A
CH B
IC1A
IC1B
IC1C
IC1D
IC1E
IC1F
40106N
VCC
VSS
VDD

A.3 Appendix: Software Modules and Paths

```
MUX_2x1              -- VHD_Source\MUX_2x1.vhd
bestMatch            -- VHD_Source\WiiVision\bestMatch.vhd
CameraXY             -- VHD_Source\WiiVision\CameraXY.vhd
CLOCKDIV             -- VHD_Source\clockdiv.vhd
CompareIt            -- VHD_Source\WiiVision\CompareIt.vhd
COORD                -- VHD_Source\Queue\coord.vhd
DIMMER               -- VHD_Source\KITTstatus\dimmer.vhd
DisplacementUnit     -- VHD_Source\Odometry\DisplacementUnit.vhd
DriveCommander       -- VHD_Source\DriveCommander\DriveCommander.vhd
GCBD_MODULE          -- VHD_Source\Odometry\gcbd_module.vhd
KITT_LIGHTS          -- VHD_Source\KITTstatus\kitt_lights.vhd
POS_CONTROL          -- VHD_Source\Queue\pos_control.vhd
PWM_FILTER           -- VHD_Source\DriveCommander\pwm_filter.vhd
PWM_MODULE           -- VHD_Source\DriveCommander\pwm_module.vhd
QUEUE                -- VHD_Source\Queue\queue.vhd
QUEUE_CONTROL        -- VHD_Source\Queue\queue_control.vhd
r2p_corproc          -- VHD_Source\CORDIC\rect2polar\r2p_corproc.vhd
sc_corproc           -- VHD_Source\CORDIC\polar2rect\sc_corproc.vhd
SteerCommander       -- VHD_Source\DriveCommander\SteerCommander.vhd
wiiCamComms          -- VHD_Source\WiiVision\wiiCamComms.vhd
WiiRefClock          -- VHD_Source\WiiVision\WiiRefClock.vhd
```

A.4 Appendix: Datasheets

PixArt Wii Remote Infrared Camera Information

Information collected from various sources for the PixArt Wii Remote Infrared Camera, since there are no publicly available datasheets.

IR Sensor

From Wiimote Wiki

Contents

Register Memory Map

Register Memory Map

0xB00000	UNKNOWN
0xB00001	0x00
0xB00002	0x00
0xB00003	0x71
0xB00004	0x01
0xB00005	0x00
0xB00006	MAXSIZE
0xB00007	0x00
0xB00008	GAIN
0xB0001A	GAINLIMIT
0xB0001B	MINSIZE
0xB00030	CONTROL
0xB00033	MODE

UNKNOWN: Wii uses values: 2 (standard) and 7 (max sensitivity). No known function.
MAXSIZE: Maximum blob size. Wii uses values from 0x62 to 0xc8.
GAIN: Sensor Gain. Smaller values = higher gain.
GAINLIMIT: Sensor Gain Limit. Must be less than GAIN for camera to function. No other effect?
MINSIZE: Minimum blob size. Wii uses values from 3 to 5
CONTROL: Write 1 before configuring camera, 8 when done.

MODE: output format, 1, 3, or 5. See output formats below.
Other values: As used by Wii. Function not known.

These registers are mirrored every 0x100 bytes up to 0xB1FFFF

Wavelengths

> The following section or page relates to parts of the Wii Remote that are poorly understood. Research and contribution in these areas is encouraged.

The sensor is at least 5 times more sensitive to 940nm than 850nm. This disparity increases slightly with the removal of the filter window. Relative sensitivity to visible is not known.

Initialization

Eight commands need to be issued to initialize the camera and request data from it. After each command, wait for the 0x22 command status report and re-issue the command if it isn't received in a reasonable time. Note that all writes are to register addresses (set 0x04 bit in first byte of the write report payload).

1. Enable IR Pixel Clock (report 0x13)
2. Enable Camera (report 0x1A)
3. Configure Camera:
    1. Write 0x01 to register 0xb00030
    2. Write configuration block 1 to registers 0xb00000-08 (9 bytes)
    3. Write configuration block 2 to registers 0xb0001A-1B (2 bytes)
    4. Write mode number to register 0xb00033
    5. Write 0x08 to register 0xb00030
4. Select output report compatable with mode (report 0x12)

Configuration

The camera configuration is somewhat unknown. Shutter time and frame rate appear to be directly determined by the pixel clock, which is an external input not controllable via software. The following settings, however, can be controlled over bluetooth:

Sensor Gain

The byte labeled GAIN controls the camera gain. Lower values result in higher sensitivity. Numerical gain is proportional to 1/2^(n/16) for n<0x40, but decreases more linearly after that. Translation: for small values, dropping this value by 0x10 doubles sensitivity. Usable values are from about 15 to 254. Below that results in noise and streaking becomes significant and the camera doesn't function for a value of 255.

A configuration byte in the second block, GAINLIMIT, must have a value less than GAIN for the camera to function. Otherwise it appears to have no effect. This byte may be related to an automatic gain control feature, but the enable for this feature has not been found.

Blob Size Range

The bytes labeled MAXSIZE and MINSIZE controls the range of a blob sizes that will be reported. The numbers appear to be approximately the number of sensor pixels in the blob blob can occupy.

For max, the Wii uses values in the range 0x62 to 0xC8, but values as low as 0x20 can be useful in specialized situations. Small values can be useful for controlling background noise and if point merging when tracking multiple points would have negative consequences. Large values allow large, unfocused, or diffuse blobs to be tracked.

For min, very small numbers work best. The Wii uses values from 3 to 5, depending on sensitivity settings.

Output Formats

Short

Output Mode value: 1

Usable with reports 0x36 and 0x37

BYTE	BIT 7	6	5	4	3	2	1	BIT 0
0	X0[7:0]							
1	Y0[7:0]							
2	Y0[9:8]		X0[9:8]		Y1[9:8]		X1[9:8]	
3	X1[7:0]							
4	Y1[7:0]							
5	X2[7:0]							
6	Y2[7:0]							
7	Y2[9:8]		X2[9:8]		Y3[9:8]		X3[9:8]	
8	X3[7:0]							
9	Y3[7:0]							

Medium

Output Mode value: 3

Usable with report 0x33

The following structure is repeated 4 times.

BYTE	BIT 7	6	5	4	3	2	1	BIT 0
0	X[7:0]							
1	Y[7:0]							
2	Y[9:8]		X[9:8]		S			

Long

Output Mode value: 5

Usable with report 0x3E/F

The following structure is repeated twice in each report.

BYTE	BIT 7	6	5	4	3	2	1	BIT 0
0	X[7:0]							
1	Y[7:0]							
2	Y[9:8]		X[9:8]		S			
3	0	XMIN						
4	0	YMIN						
5	0	XMAX						
6	0	YMAX						
7	0	0	0	0	0	0	0	0
8	INTENSITY							

Retrieved from "http://wiki.wiimoteproject.com/IR_Sensor"

Category: Unknown

---

procrastineering

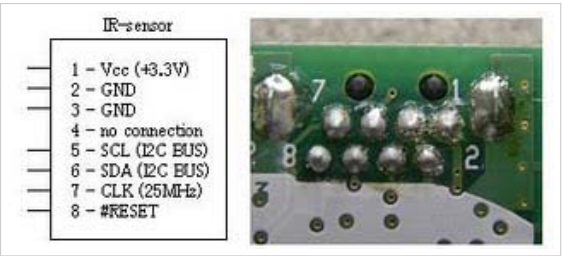
giving into productive distractions

THURSDAY, SEPTEMBER 4, 2008

Working with the PixArt camera directly

This has been a pretty whirlwind past few months. Lots of things have happened, almost none of which procrastineering related which is why I haven't posted anything here. But, one of the things that I have poked at in the past few weeks was creating a PixArt to USB-HID device which allows the camera from the Wiimote to appear as a relatively easy to access USB device. This addresses several problems with using the Wiimote such as running off batteries for extended periods and flakey platform specific Bluetooth drivers. It's also possible to read from the Pixart cam at over 100Hz if you read directly via I2C as well as track visible dots once you remove the IR filter. Of course, none of this was discovered by me. All credit belongs to the numerous individuals who have contributed thier knowledge to the various Wiimote hacking websites. Normally, this project wouldn't be worth a post, but all the information on how to do this is pretty scattered and difficult to follow. So, I figured I would contribute by trying to making this all a bit clearer.

This project is fairly advanced. You must be comfortable with working with microcontrollers. Several simpler devices such as the Arduino or the Basic Stamp may work, but I used the 18F4550 PIC Microcontroller which provides built-in full-speed USB capabilites. But first, let talk about the PixArt camera:



Here's the pinout thanks to kako and a PCB picture. The Reset pin is active low, so use a pullup resistor to Vcc. The Wiimote runs the camera with a 25Mhz clock, but it also works with a 20Mhz clock so you might get away with fudging this a bit. The I2C communication is fast 400Khz and the slave device address is 0xB0. Most microcontroller development platforms

should include I2C communication capabilities. If yours doesn't, get a better dev kit =o). Desoldering the camera can be hard with so many pins. But, careful use of a hot air gun will do the trick. The first part is to initialize the camera over I2C. Here's the pseudo code for initializing to maximum sensitivity (actual CCS C code in comments):

1. write(hex): B0 30 01
2. wait 100ms
3. write(hex): B0 00 00 00 00 00 00 00 90 //sensitivity part 1
4. wait 100ms
5. write (hex): B0 07 00 41 //sensitivity part 2
6. wait 100ms
7. write(hex): B0 1A 40 00 //sensitivity part 3
8. wait 100ms
9. write(hex): B0 33 03 //sets the mode
10. wait 100ms
11. write(hex): B0 30 08
12. wait 100ms

It's still somewhat mysterious to me what all these mean, but in this mess is the sensitivity and mode settings described at Wiibrew. The above code uses the sensitivity setting suggested by inio "00 00 00 00 00 00 90 00 41, 40 00" experssed in the 2nd, 3rd, and 4th message. The wait times are conservatively long. After you initialize, you can now read samples from it:

1. write(hex): B0 37 //prepare for reading
2. wait 25us
3. write(hex): B1 //read request
4. read 8 bytes
5. wait 380us
6. write(hex): B1 //read request
7. read 4 bytes

This yeilds one sample from the camera containing 12 bytes, 3 for each of the 4 potential points. The format of the data will be the Extended Mode (X,Y, Y 2-msb, X 2-msb, Size 4-bits). The wait timings approximate what the Wiimote does. I've called this routine 1000 times per second without ill effect. Though, I doubt this is actually scanning the sensor and instead is just reporting the contents of an interal buffer. But, people claim 200Hz updates are possible. So, you can use that as a suggestion.

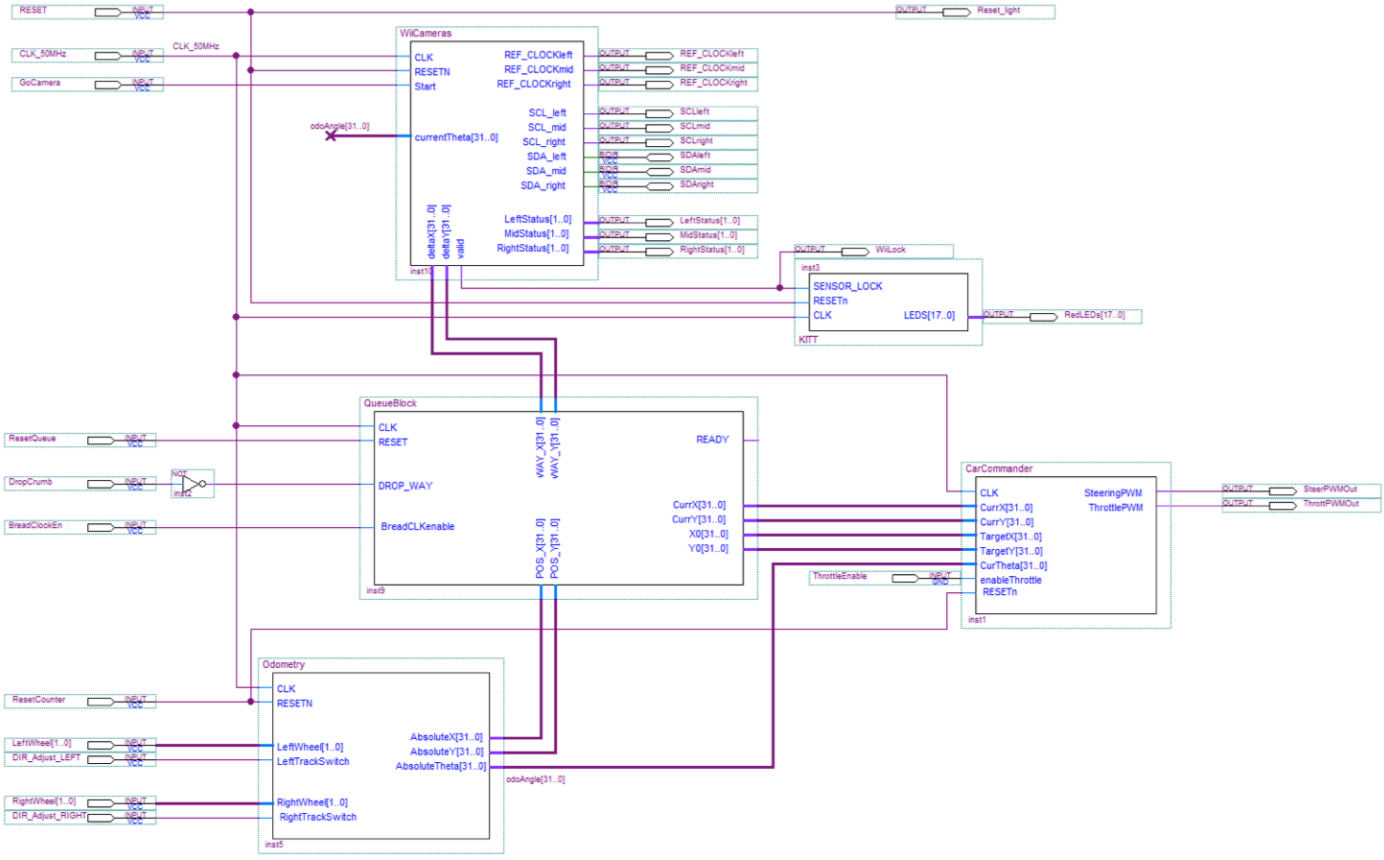
Hooking this up to your microcontroller is pretty straight forward. Give the camera 3.3v power using a voltage regulator, ground, a 20-25Mhz clock, and connect the SDA and SCL lines (don't forget your pull up resistors), and pull up the reset pin.

The CCS C Compiler for the PIC18F4550 includes USB-HID sample code. It's simply a matter of stuffing the data you got from the PixArt camera into the input report buffers

for the USB. With this, you could actually create a USB mouse profile and make it control the cursor without any software or drivers at all. If set it up as a full speed device, it's possible to get 1ms reports providing extremely low latency updates. CCS provides relatively affordable PIC programmers as well. Explaining how to set all this up is not within the scope of this post, but it should be plenty to get you started. If you want to make a PCB, you can try ExpressPCB which can get you boards in-hand for as low as $60.

Update 9/6/08: Just a note about the clock. Since my PIC was using a 20Mhz resonator, I just piggy backed the Pixart clock pin off the OSC2/CLKO pin of the PIC which seemed to work fine. Also, Kako has more details (in Japanese) on doing this with an Arduino

A.5 Appendix: Top Level Software Diagram


RESET
CLK_50MHz
GoCamera
Reset_light
WiiCameras
REF_CLOCKleft
REF_CLOCKmid
REF_CLOCKright
odoAngle[31..0]
currentTheta[31..0]
SCL_left
SCL_mid
SCL_right
SDA_left
SDA_mid
SDA_right
LeftStatus[1..0]
MidStatus[1..0]
RightStatus[1..0]
deltaX[31..0]
deltaY[31..0]
valid
WiiLock
SENSOR_LOCK
RESETn
LEDS[17..0]
RedLEDs[17..0]
KITT
QueueBlock
ResetQueue
DropCrumb
BreadClockEn
DROP_WAY
BreadCLKenable
READY
CarCommander
SteeringPWM
ThrottlePWM
SteerPWMOut
ThrottPWMOut
ThrottleEnable
Odometry
ResetCounter
LeftWheel[1..0]
DIR_Adjust_LEFT
RightWheel[1..0]
DIR_Adjust_RIGHT
AbsoluteX[31..0]
AbsoluteY[31..0]
AbsoluteTheta[31..0]