Module 2 Requirements and High Level Design

Project Requirements

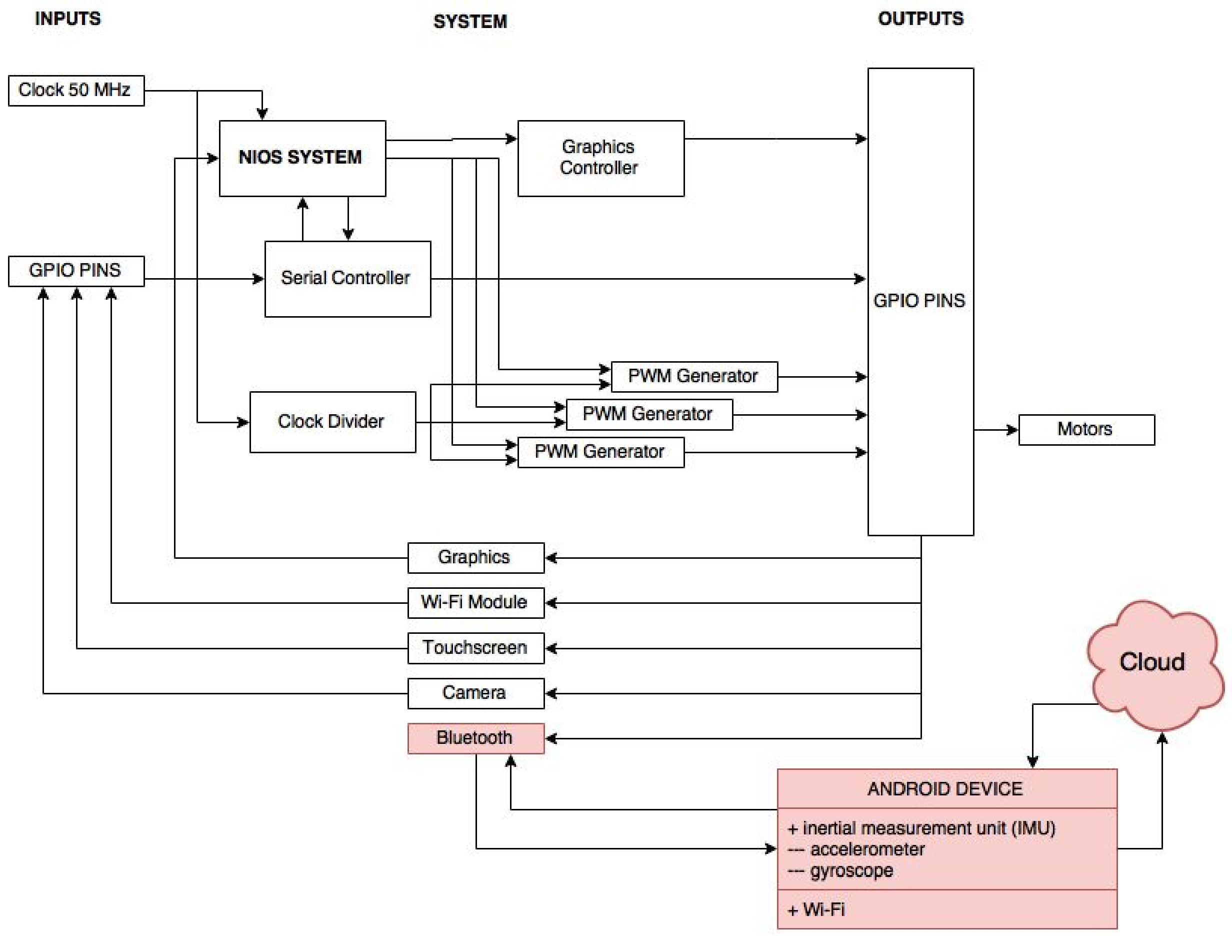
A successful application will allow a user to:

1. Control turret movements, take photos, fire with on screen buttons
2. Set the operating mode of the turret (manual, auto, security)
3. View photos taken by the turret and store them in memory
4. Press on the image to center the turret on that location
5. Set the turret to track a bright object by clicking it on screen
6. Use the phone's accelerometer to control the movements of the turret
7. Store photos taken by the turret in an online database for later viewing

High Level Design

1. Hardware Architecture

Because we are continuing our project from module 1, the system is similar to our module 1 hardware architecture but with a few extra components. Our additions are highlighted in red.



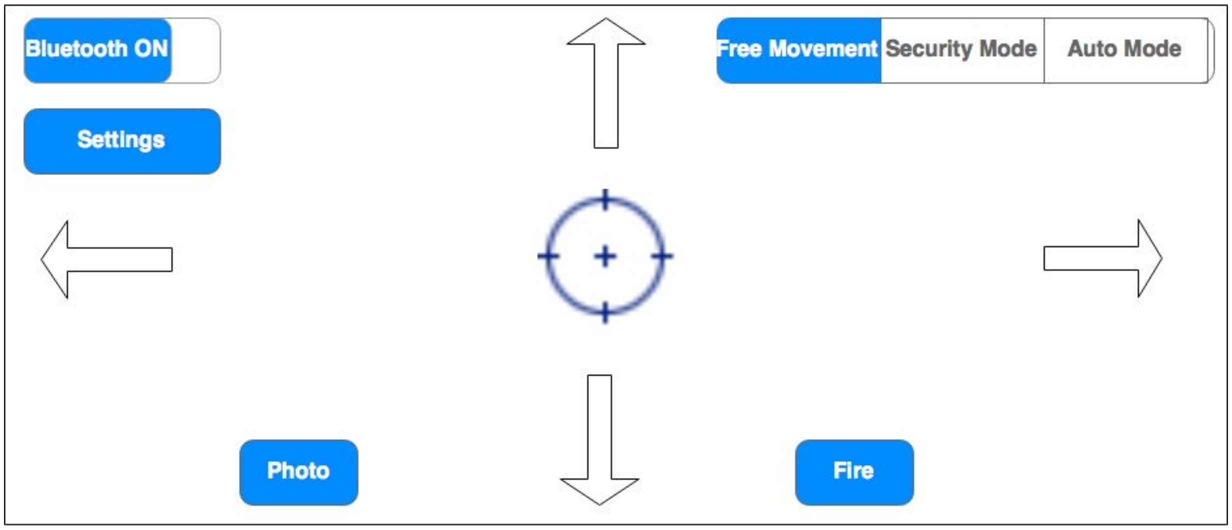
NIOS II/f: master driver that sends signals to Serial Ports and Parallel I/O Ports to communicate with other components
Motors: control movement and firing mechanism of the turret
Graphics Controller: display user interface if bluetooth is not activated
Serial Controller: deals with communication between NIOS and serial connected devices including Touchscreen, Camera, Wi-Fi Module, Bluetooth
Touchscreen: receive user input
Camera: take pictures
Wi-Fi Module: send SMS messages to user as security alerts
Bluetooth: communicates with the android device

Android Device:

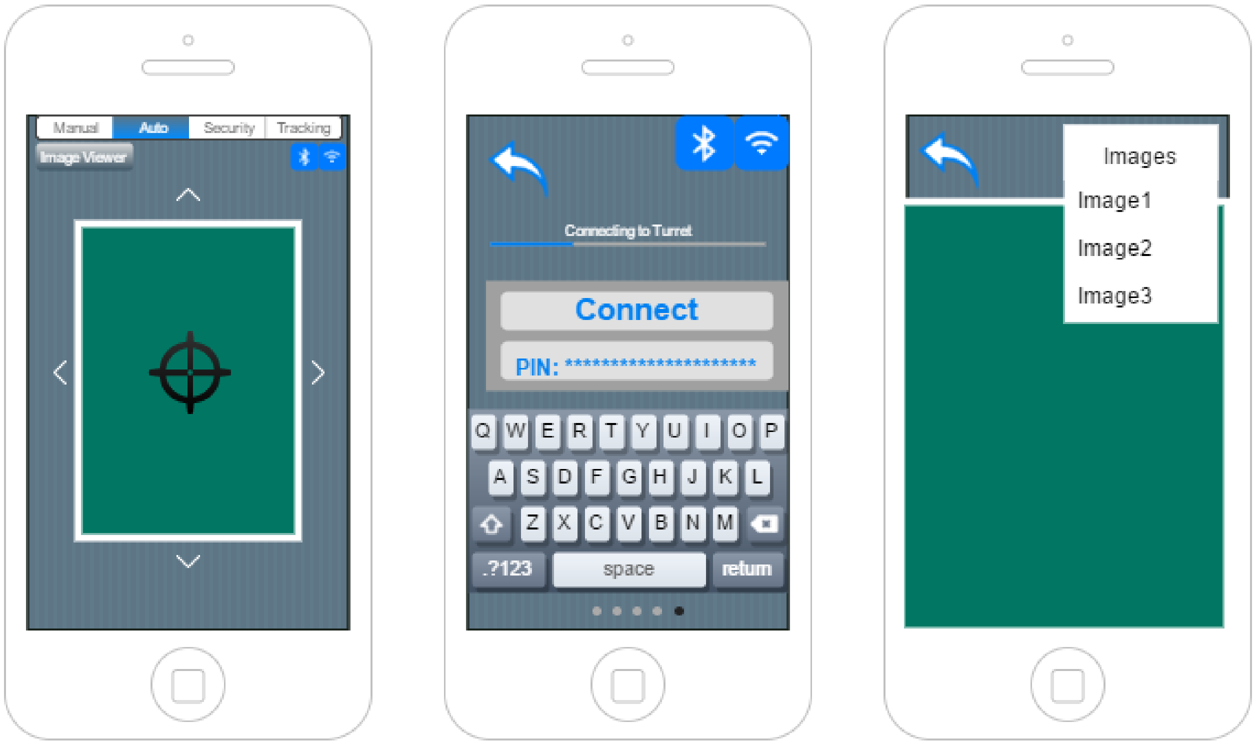
- **IMU:** detects phone movements to generate move signals to the turret
- **Wi-Fi:** send and retrieve images from the cloud

1. Graphical User Interface

NIOS II System's GUI (bluetooth button and settings buttons in addition to module 1)



Android App GUI (main screen, bluetooth connection screen, Image Viewer screen)



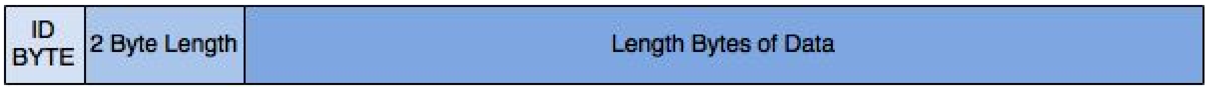
2. High Level Software Design

Communication System
Due to the limitations of NIOS II, i.e. the fact that it is single threaded and only has a 1 Byte buffer, async communications would prove very difficult, and would require complex ISRs. As such HDL1 proposes a semi-synchronous communications model. In this model, the android phone behaves as the client, and the DE1-SOC is the client.

The communication protocol used stipulates that in most cases the DE1-SOC cannot send data without having first received a request from the Android phone , similarly for all synchronous commands the android phone cannot send more data until it receives a reply or a timeout. The reasoning behind this is that due to the guarantees made by the bluetooth spec, the connection is reliable and will not drop data unexpectedly.

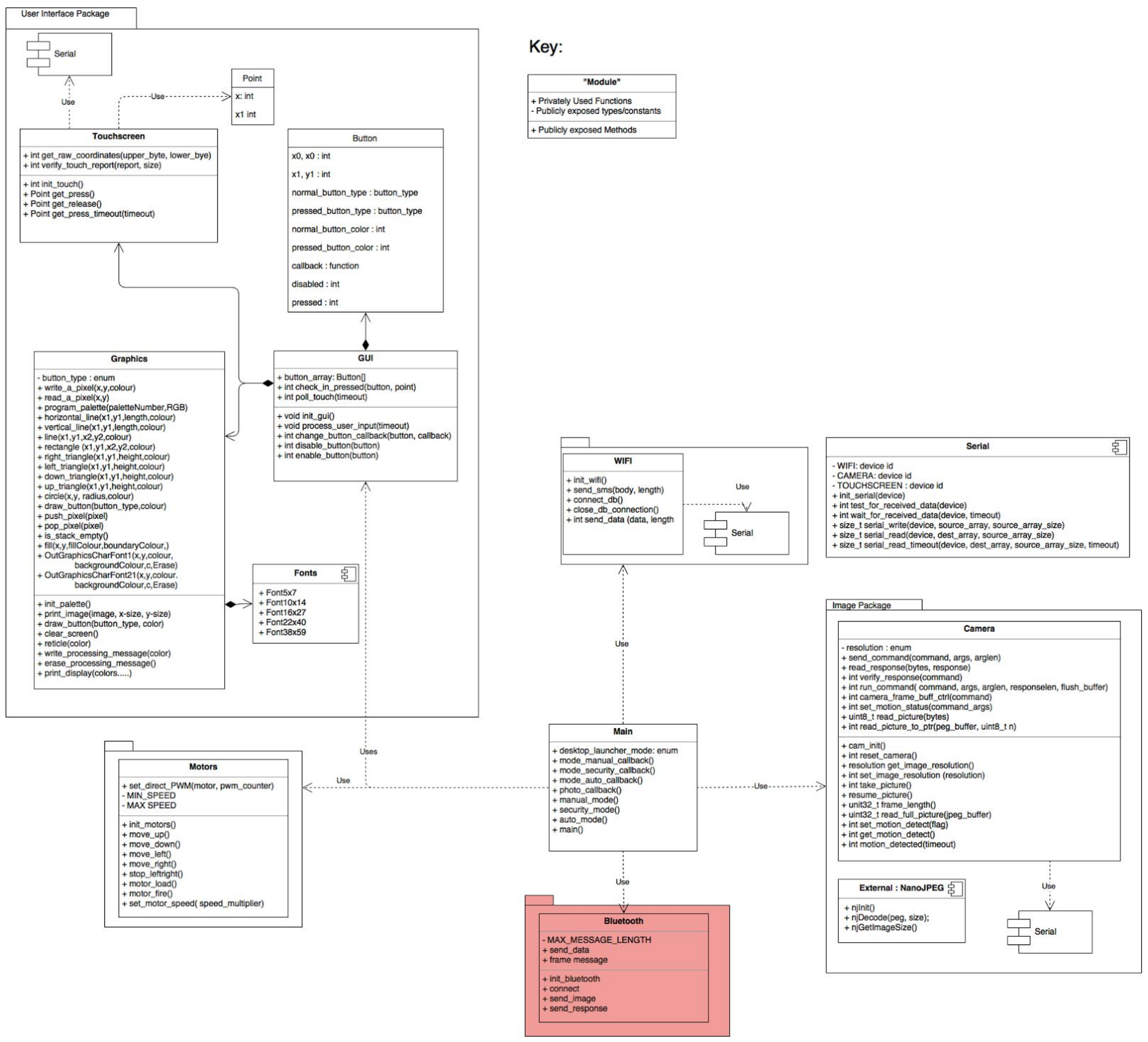
The only exception to the synchronous command rule is the motion detection message. This asynchronous message is sent from the NIOS II to the phone without any provocation, and the message will not require a response from the android phone. To implement this we would require to create a circular buffer and some basic interrupt support.

As mentioned in our report, all sent/received data will follow the following structure:



The ID identifies the message and the meaning of the data changes for each ID.
The worst case scenario is sending a JPEG which from our testing will be a max of 60K

Changes to NIOS II software



User Interface Package
Serial
Use
Use
Point
x: int
x1 int
Touchscreen
+ int get_raw_coordinates(upper_byte, lower_bye)
+ int verify_touch_report(report, size)
+ int init_touch()
+ Point get_press()
+ Point get_release()
+ Point get_press_timeout(timeout)
Button
x0, x0 : int
x1, y1 : int
normal_button_type : button_type
pressed_button_type : button_type
normal_button_color : int
pressed_button_color : int
callback : function
disabled : int
pressed : int
Graphics
- button_type : enum
+ write_a_pixel(x,y,colour)
+ read_a_pixel(x,y)
+ program_palette(paletteNumber,RGB)
+ horizontal_line(x1,y1,length,colour)
+ vertical_line(x1,y1,length,colour)
+ line(x1,y1,x2,y2,colour)
+ rectangle (x1,y1,x2,y2,colour)
+ right_triangle(x1,y1,height,colour)
+ left_triangle(x1,y1,height,colour)
+ down_triangle(x1,y1,height,colour)
+ up_triangle(x1,y1,height,colour)
+ circle(x,y, radius,colour)
+ draw_button(button_type,colour)
+ push_pixel(pixel)
+ pop_pixel(pixel)
+ is_stack_empty()
+ fill(x,y,fillColour,boundaryColour,)
+ OutGraphicsCharFont1(x,y,colour, backgroundColour,c,Erase)
+ OutGraphicsCharFont21(x,y,colour. backgroundColour,c,Erase)
+ init_palette()
+ print_image(image, x-size, y-size)
+ draw_button(button_type, color)
+ clear_screen()
+ reticle(color)
+ write_processing_message(color)
+ erase_processing_message()
+ print_display(colors.....)
GUI
+ button_array: Button[]
+ int check_in_pressed(button, point)
+ int poll_touch(timeout)
+ void init_gui()
+ void process_user_input(timeout)
+ int change_button_callback(button, callback)
+ int disable_button(button)
+ int enable_button(button)
Fonts
+ Font5x7
+ Font10x14
+ Font16x27
+ Font22x40
+ Font38x59
Key:
"Module"
+ Privately Used Functions
- Publicly exposed types/constants
+ Publicly exposed Methods
WIFI
+ init_wifi()
+ send_sms(body, length)
+ connect_db()
+ close_db_connection()
+ int send_data (data, length
Use
Serial
Serial
- WIFI: device id
- CAMERA: device id
- TOUCHSCREEN : device id
+ init_serial(device)
+ int test_for_received_data(device)
+ int wait_for_received_data(device, timeout)
+ size_t serial_write(device, source_array, source_array_size)
+ size_t serial_read(device, dest_array, source_array_size)
+ size_t serial_read_timeout(device, dest_array, source_array_size, timeout)
Use
Image Package
Camera
- resolution : enum
+ send_command(command, args, arglen)
+ read_response(bytes, response)
+ int verify_response(command)
+ int run_command(command, args, arglen, responselen, flush_buffer)
+ int camera_frame_buff_ctrl(command)
+ int set_motion_status(command_args)
+ uint8_t read_picture(bytes)
+ int read_picture_to_ptr(peg_buffer, uint8_t n)
+ cam_init()
+ int reset_camera()
+ resolution get_image_resolution()
+ int set_image_resolution (resolution)
+ int take_picture()
+ resume_picture()
+ unit32_t frame_length()
+ uint32_t read_full_picture(jpeg_buffer)
+ int set_motion_detect(flag)
+ int get_motion_detect()
+ int motion_detected(timeout)
External : NanoJPEG
+ njInit()
+ njDecode(peg, size);
+ njGetImageSize()
Use
Serial
Uses
Use
Motors
+ set_direct_PWM(motor, pwm_counter)
- MIN_SPEED
- MAX SPEED
+ init_motors()
+ move_up()
+ move_down()
+ move_left()
+ move_right()
+ stop_leftright()
+ motor_load()
+ motor_fire()
+ set_motor_speed(speed_multiplier)
Main
+ desktop_launcher_mode: enum
+ mode_manual_callback()
+ mode_security_callback()
+ mode_auto_callback()
+ photo_callback()
+ manual_mode()
+ security_mode()
+ auto_mode()
+ main()
Use
Use
Bluetooth
- MAX_MESSAGE_LENGTH
+ send_data
+ frame message
+ init_bluetooth
+ connect
+ send_image
+ send_response

We predict that twe will need to introduce another Library to the NIOS II system that handles the communication between NIOS II and the android phone. This Bluetooth Library will handle the process of framing the data, sending it, and receiving it.

Another High level change to the NIOS II system will be that there will be a mode switcher that handles enabling and disabling Bluetooth Control Mode. Additionally, the functionality of the modes will differ if the bluetooth connection is on.