Table Of Contents

1) Abstract 3
2) Executive Summary 4
3) Introduction 5
4) Integrated Systems 6
5) Mobile Platform 14
6) Actuation 15
7) Sensors 17
8) Behaviors 21
9) Conclusion 22
10) References 23
11) Appendix A: Parts and Supplies 24
12) Appendix B: Motorola 68HC12 Assembly Code 25
13) Appendix C: Pocket PC Code 38
Abstract

The goal of this project is to make an autonomous robot that can take pictures at parties of people (and/or animals). Lil’ Homie does this by using a suite of sensors for obstacle avoidance and people detection. A Motorola 68HC812 microcontroller was used in controlling actuation and getting readings from the sensor suite. An onboard Pocket PC was used to process the sensors readings, send commands to all of the connected devices (motors, servos, LEDs, and camera), house any additional data, and to provide the user with an intuitive graphical user interface. The Pocket PC also provided the user with an extra level of interactivity.
Executive Summary

Lil’ Homie is an autonomous party picture taking robot with a personality. Designed for use at parties or social gatherings, Lil’ Homie roams around a room while avoiding obstacles for a random amount of time. After that random amount, the robot stops and waits for 10 seconds for a person to jump in front of the camera. During that wait time, Lil’ Homie randomly plays one of about 10 predefined messages to call a subject over to get their picture taken. If someone comes over, there picture is taken and a praise message is played. If no one comes over, no picture is taken and a message of disapproval is played. The process begins over again until the routine is stopped.

Lil’ Homie uses two Sharp GP2D12 IR Sensors and two PVC plastic whiskers for obstacle detection. A HVW Technologies PIR Motion Sensor is used to detect people. Also, an Aiptek Mini Pencam 1.3 digital camera is used to capture the photos. Finally, a Technological Arts 68HC812 powered development board relays signals to/from motors, servos, and sensors to/from an onboard Compaq iPaq 3650 Pocket PC where the signals are processed.

The Compaq iPaq 3650 Pocket PC provides a suite of utilities to calibrate, troubleshoot, and power Lil’ Homie. The device has a 206 MHz Intel StrongArm microprocessor and 32 MB of ram which gives Lil’ Homie a tremendous amount of processing power and storage space. Also, the device gives the operator a nice user interface to work with.
Introduction

Imagine, you are in Gainesville and the Gators’ football team just won the biggest game of the season. To celebrate, you and your friends throw a 4 keg party at your place. The next morning you wake up confused on the lawn of you apartment complex, thinking to yourself, “What happened last night?” With Lil’ Homie, it’s possible to know exactly what happened last night. Also, with Lil’Homie, you can collect bribe pictures of other people doing stupid things or even get a digital guestbook of every person that walks through the door. This report describes the design process, sensor suite, and software of the robot in detail. It will also discuss pitfalls I found during the implementation of Lil’ Homie.
**Integrated Systems**

The complete block diagram of how Lil’ Homie’s systems are connected is illustrated in Figure 1. A Technological Arts Adapt812 developing board is at the heart of the bot. Every device is connected to this developing board. The microcontroller relays sensor readings to an onboard Compaq iPaq 3650 Pocket PC where all of these signals are processed and decisions are made. These decisions are sent back to the Adapt812 µP where the appropriate signals are generated for the servos, motors, and other devices. The Adapt812 only serves as a forwarding device between the Pocket PC and the connected devices; all processing and decision making is done on the Pocket PC.

![Block Diagram of Robot’s Integrated Systems](image)

**Figure 1 – Block Diagram of Robot’s Integrated Systems**

**Technological Arts Adapt812 Developing Board**

The Adapt812 Developing Board has many key features that made it, in my opinion, ideal for my robot. They are:

- **Small (2.25” x 3.25”) package**
- **90 I/O lines, all programmable as input or output, many with input capture, output compare, and "key wake-up" interrupt capability**
• 8-channel, 8-bit analog-to-digital converter

• Provided plenty of space (4K EEPROM and 1K RAM on-chip) to run robot’s slave program

• 2 SCI UART systems

The way the ports on Lil’ Homie is configured is shown in Table 1.

<table>
<thead>
<tr>
<th>Connection</th>
<th>Function</th>
<th>Port</th>
<th>Connector</th>
<th>Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Servo 1</td>
<td>Pan left/right movement</td>
<td>PT3/OC3</td>
<td>H1</td>
<td>10</td>
</tr>
<tr>
<td>Servo 2</td>
<td>Tilt up/down movement</td>
<td>PT2/OC2</td>
<td>H1</td>
<td>11</td>
</tr>
<tr>
<td>PWM1</td>
<td>Left Motor Speed Control Signal</td>
<td>PT0/OC0</td>
<td>H1</td>
<td>13</td>
</tr>
<tr>
<td>PWM2</td>
<td>Right Motor Speed Control Signal</td>
<td>PT1/OC1</td>
<td>H1</td>
<td>12</td>
</tr>
<tr>
<td>1A</td>
<td>Motor Direction Control</td>
<td>PJ3/KWJ3</td>
<td>H1</td>
<td>18</td>
</tr>
<tr>
<td>2A</td>
<td>Motor Direction Control</td>
<td>PJ2/KWJ2</td>
<td>H1</td>
<td>19</td>
</tr>
<tr>
<td>3A</td>
<td>Motor Direction Control</td>
<td>PJ1/KWJ1</td>
<td>H1</td>
<td>20</td>
</tr>
<tr>
<td>4A</td>
<td>Motor Direction Control</td>
<td>PJ0/KWJ0</td>
<td>H1</td>
<td>21</td>
</tr>
<tr>
<td>Left Front IR</td>
<td>Left IR Readings</td>
<td>PAD0/AN0</td>
<td>H1</td>
<td>22</td>
</tr>
<tr>
<td>Right Front IR</td>
<td>Right IR Readings</td>
<td>PAD1/AN1</td>
<td>H1</td>
<td>23</td>
</tr>
<tr>
<td>Left Bump</td>
<td>Left Bump Whisker</td>
<td>ADDR17/PG1</td>
<td>H2</td>
<td>37</td>
</tr>
<tr>
<td>Right Bump</td>
<td>Right Bump Whisker</td>
<td>ADDR16/PG0</td>
<td>H2</td>
<td>36</td>
</tr>
<tr>
<td>Camera Mode Relay</td>
<td>Mode</td>
<td>ADDR9/PA1</td>
<td>H2</td>
<td>32</td>
</tr>
<tr>
<td>Camera Shutter Relay</td>
<td>Shutter</td>
<td>ADDR8/PA0</td>
<td>H2</td>
<td>33</td>
</tr>
</tbody>
</table>

Table 1 – Connections made on Adapt812
The protocol created for the Pocket PC and the microcontroller to communicate with each other is quite simple. The communication algorithm is illustrated in the flow chart in Figure 3. Basically, the microcontroller constantly polls to see if any new data has been received. Once data is received, a comparison is made to the byte that is stored in the receive buffer. If the byte is equal to 0₂, then the following byte is the value of the speed for the left motor. If the byte is equal to 1₂, then the following byte is the value of the speed for the right motor. The protocol pretty much follows this same routine to control the pan-and-tilt servos, turn camera on, take picture, and change direction of the motors (see flow chart). However, when the byte in the receive buffer is equal to 5₂, the microcontroller sends all of the sensor readings to the Pocket PC which triggers an interrupt on the Pocket PC so it can handle the data.
Received Byte

Status Flag

=0

=1

Status Flag

=0

=1

Grab Byte in receive buffer. Compute PWM value from byte received. Store value in OC register for LEFT Motor.

Grab Byte in receive buffer. Compute PWM value from byte received. Store value in OC register for RIGHT Motor.

Grab Byte in receive buffer.

Received Byte

=0

=1

Set Motors to Forward.

Set Motors to Reverse.

Start

=3

=4

=5

Start

Grab Byte in receive buffer. Compute PWM value from byte received. Store value in OC register for UP_DOWN servo

Grab Byte in receive buffer. Compute PWM value from byte received. Store value in OC register for UP_DOWN servo

Send IR Right. Send IR Left. Send Bump. Send Pyro

Start
Compaq iPaq 3650 Pocket PC

An onboard Compaq iPaq 3650 Pocket PC is used for processing all of the data received from all of the sensors, providing the user with an intuitive user interface, and executing Lil’ Homie’s behaviors. Having the Pocket PC implemented is good for many reasons:

- **Makes Lil’ Homie’s platform completely universal; that is, many applications can be created and stored on the Pocket PC for the robot to run.**
- **Monster processing power is available.** Normally, having a large amount of processing power is only available by transmitting video or data to a distant laptop or workstation. With the new generation of Pocket PCs, up to 400 MHz is available in a small package. Lil’ Homie’s Pocket PC’s clock rate is at 206 MHz.
- **Lots of storage for program code and data.** Normally today’s Pocket PCs have about 64MB of ram with the option to have up to 1 GB of CompactFlash storage space. Lil’ Homie has 32 MB ram which seems to be plenty for most applications.
- **Intuitive User Interface.** There are many applications for robots that could help with a person’s busy life. Since Pocket PCs have an easy to use Windows environment (with the option to have Linux instead if desired), the complexity of operating a robot is eliminated.

The PPC software suite for Lil’ Homie includes a Main Menu that links many utilities for troubleshooting and calibration of the bot. You can also execute Lil’ Homie’s main behavior from this Main Menu. Lil’ Homie’s Main Menu is pictured in Figure 3.

---

**Figure 3 – Complete Flow Chart of uP and PPC Protocol**
There are four functions currently implemented for Lil’ Homie—the Control Panel, Actuation Calibration, Test Camera, and Take Pictures.

**Control Panel**

The control panel is a where you can easily see potential problems with Lil’ Homie’s sensor and actuator suite. From the main menu, you can see the value readings for each sensor, manually control each motor, run an Obstacle Avoidance behavior program, and manually control each servo. You also see exactly what values are stored in the servos’ output compare registers to generate the appropriate waveform for the servos’ position.
In testing motors, I found that Lil’ Homie moves differently depending on what surface it travels on. In the actuation calibration menu, the user can alter what values the motors get for the platform to travel straight or to turn on different surfaces.

![Little Homie Actuation Calibration Menu](image)

**Figure 6 – Lil’ Homie’s Actuation Calibration Menu**

**Test Camera**

In the test camera menu, it is possible to turn on the Aiptek digital camera, take a picture, and erase the contents of the camera.

![Camera Control Panel](image)

**Figure 7 – Lil’ Homies Camera Control Panel**

**Take Pictures!**

The take pictures function is the main function of the bot. This is the area to execute the Take Pictures program. When the robot is ready to start the routine, the “Start Taking Picture” button
becomes enabled. Whenever an important event occurs, a message is written in the messages text box.

Figure 8 – Lil’ Homie’s Main Program
**Mobile Platform**

One of the things that I would most like to redesign is Lil’ Homie’s platform. As shown in figure 9, the platform is mostly rectangle with two DC motors used for actuation. A rear ball caster is used to hold the back end off the ground. An acrylic box was constructed to house all of the electronics and to serve as a platform for the pan-and-tilt “head” and PPC. On the underbelly, 12 AA batteries supply power to the motors and electronics.

The main reason why I would like to redesign the platform is that it is too small. After adding the microcontroller board, breadboard, pan-and-tilt, and PPC, there is no room left to add other devices.

![Figure 9 – Lil’ Homie](image)
**Actuation**

Two Tamiya High Power Gearbox kits were used for actuation of the platform. They are set to a gear aspect ratio of 68.4:1 to get a torque of 1040 g-cm. At 4.5V, they run at about 160 RPM. When the motors are stalled, they pull a little over 1A (more like 1.15A) each. One TI SN754410 motor driver controls the motors. The wiring diagram is shown in Figure 10. The motor driver’s logic table is shown in Table 2.

![Wiring diagram of Lil' Homie's motor driver](image)

**Figure 10 – Wiring diagram of Lil’ Homie’s motor driver**

<table>
<thead>
<tr>
<th>PWM1</th>
<th>1A</th>
<th>2A</th>
<th>Motor Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>-</td>
<td>Coast</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Dynamic Braking</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Dynamic Braking</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Motor Forward</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Motor Reverse</td>
</tr>
</tbody>
</table>

**Table 2 – Example logic table for M1**
In addition to the two DC motors, there are two Hitec 300 servos controlling the pan-and-tilt mechanism for the camera.
Sensors

Lil’ Homie has two Sharp GP2D12 IR Sensors and two PVC whisker bump switches in the front for obstacle avoidance, a HVW Technologies PIR Motion Detector for detecting people, and an Aiptek Mini Pencam 1.3 digital camera to capture pictures with.

Sharp GP2D12 IR Sensors

The Sharp IR Sensors give analog readings from 1 – 255, where 1 means there is no object detected and 255 means there is an object very close. The sensors were mounted on Lil’ Homie as shown in Figure 2.

![Figure 2 – IR arrangement](image)

HVW Tech PIR Motion Detector

The HVW Tech PIR Motion Detector is a simple and cheap pyroelectric sensor that detects the movement of the inferred radiation signature of humans and animals. The detector has a simple three line setup as shown figure 3. It takes about 1 min, once Lil’ Homie is first powered on, for the pyroelectric sensor to stabilize and get readings.
One very important thing I noticed is that if the platform that the detector sits on moves, the PIR sensor generates unreliable readings. As shown in Figure 4, the readings from the sensor bounces from +5V to 0V many times before it settles down to give accurate readings. It seems that the pyro sensor takes no more than 5 seconds to settle once the platform stops moving. To overcome this, Lil’ Homie has to wait for those 5 seconds before polling to see if a person is detected.

Figure 4 – Example Timing Diagram on PIR when platform moves

Aiptek Mini Pencam 1.3

Originally, the CMUcam was going to be used to capture the photos. The photos then would have been sent to the Pocket PC where they would be stored. Unfortunately, the CMUcam doesn’t take
acceptable pictures for this application; instead, the CMUcam should be only used for applications in computer vision.

To get around the problem, I hacked into a cheap ~$50 Aiptek Mini Pencam 1.3 digital camera. The key features of this camera are:

- Small $1^\prime\times1^\prime\times3\frac{1}{2}^\prime$ package
- 32 MB flash can store 140 images at maximum
- USB interface
- Low Powered – runs off only two AAA batteries which can take about 140 pictures on
- 1.3 MPixel CMOS Sensor takes images at 1248 x 960 pixels
- Can take video
- Simple two button operation consists of a Mode and Shutter button

To hack it, wires had to be soldered from the Mode and Shutter buttons to two separate relay switches that the microcontroller could control. Figures 5 and 6 show how the wires are soldered on. Figure 7 shows the circuit that constructed to get the microcontroller to control the Mode and Shutter features.

Figure 5 – Mode Button on Pencam
Figure 6 – Shutter Button on Pencam

Figure 7 – Wiring to from Pencam to relay
**Behaviors**

Lil’ Homie has four behaviors—that is, obstacle avoid, call a person over, wait for person, and take a picture. For a random amount of time between 5 to 30 seconds, Lil’ Homie roams around a room while avoiding obstacles. Once that time has elapsed, Lil’ Homie plays a 5 second message to call a person over. While the message plays, the pyro sensor stabilizes its readings. Next, Lil’ Homie waits another 5 seconds for a person to come within the view of the camera. If a person is detected, a picture is taken and a praise message is played. If a person is not detected, Lil’ Homie tells everybody that they are no fun. Finally, the process begins all over again with Lil’ Homie obstacle avoiding until the routine is stopped.
Conclusion

In summary, Lil’ Homie was somewhat of a success. The robot actually does take good pictures of people at parties. Interfacing the Pocket PC proved to be very beneficial in troubleshooting problems found in the connected devices. It also became a great tool for conducting experiments.

The next plan for Lil’ Homie is to interface the PPC to the camera in such a way that the pictures taken can be viewed on the PPC. This will be an extremely hard thing to achieve because of 1) connectivity issues and 2) driver issues. Although this is true, I believe that it is achievable.

In building Lil’ Homie, I learned more things than I’ve ever learned while attending UF. IMDL proved to be an expensive but the most rewarding experience I’ve ever encountered. I’ve now grown to have an extreme interest in robotics.
References


Appendix A: Parts and Supplies

Sharp GP2D12 Distance Measuring Sensor  
MarK III Robot Store  
2889 Tolkien Lane  
Lake Oswego, OR 97034  
http://www.junun.org/MarkIII/  
(503)638-8407

BMP-01 Bumper Switches  
Lynxmotion, Inc.  
PO Box 812  
Pekin, IL 61554-0818  
http://www.lynxmotion.com  
(309)382-1816

PIR Motion Detector  
HVW Technologies Inc.  
218, 3907 – 3A St. N.E.  
Calgary, Alberta T2E 6S7  
CANADA  
http://www.HVWTech.com  
(403)730-8603

Tamiya 70144 Ball Caster Kit  
Pololu Corporation  
3335 Hauck St. #1023  
Las Vegas, NV 89146  
http://www.pololu.com  
1-8777-7-POLOLU

Hitech 300 Servos  
Tamiya High Powered Motor Kit  
Mondo-Tronics  
124 Paul Drive, Suite 12  
San Rafael, CA 9493  
http://www.robotstore.com  
(415)491-4600

Aiptek Mega Pencam 1.3  
Wal-Mart  
3570 SW Archer Rd  
Gainesville, FL 32608  
(352) 371-3171
Appendix B – Motorola 68HC12 Assembly Code

/*********************************** *
* Slave Code for Lil’Homie Robot
* PROGRAMMER: Jesse Martin
* UPDATED: December 9, 2002
*************************************/

* Operational Parameters

_100MS       equ     25
_250MS       equ     61
_500MS       equ     125
_1SECOND     equ     250
_2SECONDS    equ     500
_3SECONDS    equ     750
_5SECONDS    equ     1250
_10SECONDS   equ     2500
_25SECONDS   equ     6250
_60SECONDS   equ     15000
_2MIN        equ     30000

RAM      equ     $0800           ;68HC812A4 internal RAM
STACK    equ     $0bff           ;Stack at top of internal ram
rbase    equ     $0000           ;68HC812A4 register block
EEPROM   equ     $f000           ;68HC812A4 internal EEPROM
CONSTANTS equ $fe00  ;CONSTANTS STORED IN EEPROM

* Operational Constants

TRUE            equ     $FF
FALSE           equ     $00
CR              equ     $D
LF              equ     $A
SPACE           equ     $20

;RTI Variables
clrmask         equ     %11000000       ;mask for clearing timer flags
rtimask1        equ     %10000001       ;M=8Mhz, 1.024 msec interrupt with 16 MHz
xtal
rtimask2        equ     %10000010       ;M=8Mhz, 2.048 msec interrupt with 16 MHz
xtal
rtimask3        equ     %10000011       ;M=8Mhz, 4.096 msec interrupt with 16 MHz
xtal
rtimask4        equ     %10000100       ;M=8Mhz, 8.192 msec interrupt with 16 MHz
xtal
rtimask5        equ     %10000101       ;M=8Mhz, 16.384 msec interrupt with 16 MHz
xtal
rtimask6        equ     %10000110       ;M=8Mhz, 32.768 msec interrupt with 16 MHz
xtal
rtimask7        equ     %10000111       ;M=8Mhz, 65.536 msec interrupt with 16 MHz
xtal
rtiflag equ %10000000

;SCI Variables
scimask equ %00101100 ;RIE - SCI Interrupt enable
RE - Receiver Enable
RDRFflag equ %00100000 ;RDRF - Receive Data Register Full flag
TDREflag equ %10000000 ;TDRE - Transmit Data Register Empty flag

;Baud rate definitions
;MCLK=8Mzh
BAUD110 equ 4545 ;(baud) 110 baud with 16 Mhz crystal
BAUD300 equ 1667 ;(baud) 300 baud with 16 Mhz crystal
BAUD600 equ 833 ;(baud) 600 baud with 16 Mhz crystal
BAUD1200 equ 417 ;(baud) 1200 baud with 16 Mhz crystal
BAUD2400 equ 208 ;(baud) 2400 baud with 16 Mhz crystal
BAUD4800 equ 104 ;(baud) 4800 baud with 16 Mhz crystal
BAUD9600 equ 52 ;(baud) 9600 baud with 16 Mhz crystal
BAUD14400 equ 35 ;(baud) 14400 baud with 16 Mhz crystal
BAUD19200 equ 26 ;(baud) 19200 baud with 16 Mhz crystal
BAUD38400 equ 13 ;(baud) 38400 baud with 16 Mhz crystal

* Registers
REG EQU $0000

PORTA EQU $0000 ;PORTA
PORTB EQU $0001 ;PORTB
DDRA EQU $0002 ;PORTA - DATA DIRECTION REGISTER
DDRB EQU $0003 ;PORTB - DATA DIRECTION REGISTER
PORTC EQU $0004 ;PORTC
PORTD EQU $0005 ;PORTD
DDRC EQU $0006 ;PORTC - DATA DIRECTION REGISTER
DDRD EQU $0007 ;PORTD - DATA DIRECTION REGISTER
PORTE EQU $0008 ;PORTE
DDRE EQU $0009 ;PORTE - DATA DIRECTION REGISTER
PEAR EQU $000A ;PEAR - PORTE ASSIGNMENT REGISTER
MODE EQU $000B ;MODE - MODE REGISTER
PUCR EQU $000C ;PUCR - PULL UP CONTROL REGISTER
RDRIV EQU $000D ;RDRIV - REDUCED DRIVE OF I/O LINES

INITRM EQU $0010 ;INITRM - INITIALIZATION OF INTERNAL RAM POSITION REGISTER
INITRG EQU $0011 ;INITRG - INITIALIZATION OF INTERNAL REGISTER POSITION REGISTER
INITEE EQU $0012 ;INITEE - INITIALIZATION OF INTERNAL EEPROM POSITION REGISTER
MISC EQU $0013 ;MISC - MISCELLANEOUS MAPPING CONTROL REGISTER
RTICTL EQU $0014 ;RTICTL - REAL TIME INTERRUPT CONTROL REGISTER
RTIFLG EQU $0015 ;RTIFLG - REAL TIME INTERRUPT FLAG REGISTER
COPCTL EQU $0016 ;COPCTL - COP CONTROL REGISTER
COPRST EQU $0017 ;COPRST - ARM/RESET COP TIMER REGISTER
ITST0 EQU $0018 ;ITST0
ITST1 EQU $0019 ;ITST1
ITST2 EQU $001A ;ITST2
ITST3 EQU $001B ;ITST3
INTCR EQU $001E ;INTCR - INTERRUPT CONTROL REGISTER
HPRIO EQU $001F ;HPRIO - HIGHEST PRIORITY INTERRUPT
KWIED EQU $0020 ;KWIED - KEY WAKEUP PORTD INTERRUPT ENABLE REGISTER
KWIFD EQU $0021 ;KWIFD - KEY WAKEUP PORTD FLAG REGISTER

PORTH EQU $0024 ;PORTH
DDRH EQU $0025 ;DDRH - DATA DIRECTION REGISTER
KWIEH EQU $0026 ;KWIEH - KEY WAKEUP PORTH INTERRUPT ENABLE REGISTER
KWIFH EQU $0027 ;KWIFH - KEY WAKEUP PORTH FLAG REGISTER
PORTJ EQU $0028 ;PORTJ
DDRJ EQU $0029 ;DDRJ DATA DIRECTION REGISTER
KWIEJ EQU $002A ;KWIEJ - KEY WAKEUP PORTJ INTERRUPT ENABLE REGISTER
KWIFJ EQU $002B ;KWIFJ - KEY WAKEUP PORTJ FLAG REGISTER
KPOLJ EQU $002C ;KPOLJ - KEY WAKEUP PORTJ POLARITY REGISTER
PUPSJ EQU $002D ;PUPSJ - KEY WAKEUP PORTJ PULL-UP/PULLDOWN SELECT REGISTER
PULEJ EQU $002E ;PULEJ - KEY WAKEUP PORTJ PULL-UP/PULLDOWN ENABLE REGISTER

PORTF EQU $0030 ;PORTF
PORTG EQU $0031 ;PORTG
DDRF EQU $0032 ;DDRF - DATA DIRECTION REGISTER
DDRG EQU $0033 ;DDRG - DATA DIRECTION REGISTER
DPAGE EQU $0034 ;DPAGE - DATA PAGE REGISTER
PPAGE EQU $0035 ;PPAGE - PROGRAM PAGE REGISTER
EPAGE EQU $0036 ;EPAGE - EXTRA PAGE REGISTER
WINDEF EQU $0037 ;WINDEF - WINDOW DEFINITION REGISTER
MXAR EQU $0038 ;MXAR - MEMORY EXPANSION ASSIGNMENT REGISTER

CSCTL0 EQU $003C ;CSCTL0 - CHIP SELECT CONTROL REGISTER 0
CSCTL1 EQU $003D ;CSCTL1 - CHIP SELECT CONTROL REGISTER 1
CSSTR0 EQU $003E ;CSSTR0 - CHIP SELECT STRETCH REGISTER 0
CSSTR1 EQU $003F ;CSSTR1 - CHIP SELECT STRETCH REGISTER 1
LDVH EQU $0040 ;LDV - LOOP DIVIDER HIGH REGISTER
LDVL EQU $0041 ;LDV - LOW REGISTER
RDVH EQU $0042 ;RDV - REFERENCE DIVIDER HIGH REGISTER
RDVL EQU $0043 ;RDV - LOW REGISTER

CLKCTL EQU $0047 ;CLKCTL - CLOCK CONTROL REGISTER

ATDCTL0 EQU $0060 ;ATDCTL0 - RESERVED
ATDCTL1 EQU $0061 ;ATDCTL1 - RESERVED
ATDCTL2 EQU $0062 ;ATDCTL2 - ATD CONTROL REGISTER
ATDCTL3 EQU $0063 ;ATDCTL3 - ATD CONTROL REGISTER
ATDCTL4 EQU $0064 ;ATDCTL4 - ATD CONTROL REGISTER
ATDCTL5 EQU $0065 ;ATDCTL5 - ATD CONTROL REGISTER

ATDSTAT EQU $0066 ;ATDSTAT - ATD STATUS HIGH REGISTER
ATDSTATH EQU $0066 ;ATDSTAT - ATD STATUS HIGH REGISTER
ATDSTATL EQU $0067 ;ATDSTAT - LOW REGISTER

STDTEST EQU $0068 ;STDTEST - ATD TEST HIGH REGISTER
ATDTESTH EQU $0068 ;ATDTEST - ATD TEST HIGH REGISTER
ATDTESTL EQU $0069 ;ATDTEST - ATD TEST LOW REGISTER
PORTAD EQU $006F ;PORTAD - PORT AD DATA INPUT REGISTER
ADR0H EQU $0070 ;ADR0H
ADR1H EQU $0072 ;ADR1H
ADR2H EQU $0074 ;ADR2H
ADR3H EQU $0076 ;ADR3H
ADR4H EQU $0078 ;ADR4H
ADR5H EQU $007A ;ADR5H
ADR6H EQU $007C ;ADR6H
ADR7H EQU $007E ;ADR7H
TIOS EQU $0080 ;TIOS - TIMER INPUT CAPTURE/OUTPUT COMPARE SELECT
CFORC EQU $0081 ;CFORC - TIMER COMPARE FORCE REGISTER
OC7M EQU $0082 ;OC7M - OUTPUT COMPARE 7 MASK REGISTER
OC7D EQU $0083 ;OC7D - OUTPUT COMPARE 7 DATA REGISTER
TCNT EQU $0084
TCNTH EQU $0084 ;TCNT - TIMER COUNT HIGH REGISTER
TCNTL EQU $0085 ;TCNT - HIGH REGISTER
TSCR EQU $0086 ;TSCR - TIMER SYSTEM CONTROL REGISTER
TQCR EQU $0087 ;TQCR - RESERVED
TCTL1 EQU $0088 ;TCTL1 - TIMER CONTROL REGISTER 1
TCTL2 EQU $0089 ;TCTL2 - TIMER CONTROL REGISTER 2
TCTL3 EQU $008A ;TCTL3 - TIMER CONTROL REGISTER 3
TCTL4 EQU $008B ;TCTL4 - TIMER CONTROL REGISTER 4
TMSK1 EQU $008C ;TMSK1 - TIMER INTERRUPT MASK 1
TMSK2 EQU $008D ;TMSK2 - TIMER INTERRUPT MASK 2
TFLG1 EQU $008E ;TFLG1 - TIMER INTERRUPT FLAG 1
TFLG2 EQU $008F ;TFLG2 - TIMER INTERRUPT FLAG2
TC0 EQU $0090
TC0H EQU $0090 ;TC0 - TIMER INPUT/CAPTURE COMPARE HIGH REGISTER
TC0L EQU $0091 ;TC0 - LOW REGISTER
TC1 EQU $0092
TC1H EQU $0092 ;TC1 - HIGH REGISTER
TC1L EQU $0093 ;TC1 - LOW REGISTER
TC2 EQU $0094
TC2H EQU $0094 ;TC2 - HIGH REGISTER
TC2L EQU $0095 ;TC2 - LOW REGISTER
TC3 EQU $0096
TC3H EQU $0096 ;TC3 - HIGH REGISTER
TC3L EQU $0097 ;TC3 - LOW REGISTER
TC4 EQU $0098
TC4H EQU $0098 ;TC4 - HIGH REGISTER
TC4L EQU $0099 ;TC4 - LOW REGISTER
TC5 EQU $009A
TC5H EQU $009A ;TC5 - HIGH REGISTER
TC5L EQU $009B ;TC5 - LOW REGISTER
TC6 EQU $009C
TC6H EQU $009C ;TC6 - HIGH REGISTER
TC6L EQU $009D ; TC6 - LOW REGISTER
TC7 EQU $009E
TC7H EQU $009E ; TC7 - HIGH REGISTER
TC7L EQU $009F ; TC7 - LOW REGISTER
PACTL EQU $00A0 ; PATCL - PULSE ACCUMULATOR CONTROL REGISTER
PAFLG EQU $00A1 ; PAFLG - PULSE ACCUMULATOR FLAG REGISTER
PACNT EQU $00A2 ; PACNT - 16 BIT PULSE ACCUMULATOR COUNT REGISTER
PACNTL EQU $00A3 ; PACNT - LOW REGISTER
TMTST EQU $00AD ; TMTST - TIMER TEST REGISTER
PORTT EQU $00AE ; PORTT
DDRT EQU $00AF ; DDRT - DATA DIRECTION REGISTER
SC0BDH EQU $00C0 ; SC0BDH - SCI BAUD RATE CONTROL REGISTER
SC0BDL EQU $00C1 ; SC0BDL - SCI BAUD RATE CONTROL REGISTER
SC0CR1 EQU $00C2 ; SC0CR1 - SCI CONTROL REGISTER
SC0CR2 EQU $00C3 ; SC0CR2 - SCI CONTROL REGISTER
SC0SR1 EQU $00C4 ; SC0SR1 - SCI STATUS REGISTER
SC0SR2 EQU $00C5 ; SC0SR2 - SCI STATUS REGISTER
SC0DRH EQU $00C6 ; SC0DRH - SCI DATA REGISTER
SC0DRL EQU $00C7 ; SC0DRL - SCI DATA REGISTER
SC1BDH EQU $00C8 ; SC1BDH - SCI BAUD RATE CONTROL REGISTER
SC1BDL EQU $00C9 ; SC1BDL - SCI BAUD RATE CONTROL REGISTER
SC1CR1 EQU $00CA ; SC1CR1 - SCI BAUD CONTROL REGISTER
SC1CR2 EQU $00CB ; SC1CR2 - SCI CONTROL REGISTER
SC1SR1 EQU $00CC ; SC1SR1 - SCI STATUS REGISTER
SC1SR2 EQU $00CD ; SC1SR2 - SCI STATUS REGISTER
SC1DRH EQU $00CE ; SC1DRH - SCI DATA REGISTER
SC1DRL EQU $00CF ; SC1DRL - SCI DATA REGISTER
SP0CR1 EQU $00D0 ; SP0CR1 - SPI CONTROL REGISTER
SP0CR2 EQU $00D1 ; SP0CR2 - SPI CONTROL REGISTER
SP0BR EQU $00D2 ; SP0BR - SPI BAUD RATE REGISTER
SP0SR EQU $00D3 ; SP0SR - SPI STATUS REGISTER
SP0DR EQU $00D5 ; SP0DR - SPI DATA REGISTER
PORTS EQU $00D6 ; PORTS
DDRS EQU $00D7 ; DDRS - DATA DIRECTION REGISTER
EECHR EQU $00F0 ; EECHR - EEPROM MODULE CONFIGURATION
EECHR EQU $00F1 ; EECHR - EEPROM BLOCK PROTECT
EETST EQU $00F2 ; EETST - EEPROM TEST
EECHR EQU $00F3 ; EECHR - EEPROM CONTROL

*MASKS
BIT0 EQU %00000001
BIT1 EQU %00000010
BIT5 EQU %00100000
BIT7 EQU %10000000
BIT76 EQU %11000000
BIT41 EQU %00010010
BIT32 EQU %00001100
; ORG CONSTANTS
;MENU DC.B "SELECT AN OPTION:", LF, "1)TURN SERVO LEFT", LF, "2)TURN SERVO RIGHT", LF, "3)CENTER SERVO", 0

************************************************
* MAIN PROGRAM
************************************************
ORG EEPROM
LDS #STACK
JSR INIT_SCI ;INIT SCI SYSTEM
JSR INIT_MOTOR ;INIT MOTORS
JSR INIT_SERVOS ;INIT SERVOS
JSR INIT_AD ;INIT AD SYSTEM
CLI

;Initialize PORTT - LED on board
movb #%01100000, DDRT ;Bit 6,5 of PORTT are Output
bclr PORTT, %01100000

;Initialize PORTA
movb #%00000011, DDRA ;Bit 0,1 of PORTA are Output
bclr PORTT, %00000011

;Motors forward
movb #%00000101, PORTJ

LOOP LDAA SC0SR1 ; check status reg (RDRF in bit 5)
       ANDA #%00100000 ; check if receive buffer full
       BEQ LOOP ; wait until data present
       LDAB SC0DRL
       CMPB #0 ; first byte = 0 means following byte is for left
       BEQ LEFT_MOTOR
       CMPB #1 ; first byte = 1 means following byte is for right
       BEQ RIGHT_MOTOR
       CMPB #2 ; first byte = 2 means following byte is direction
       BEQ DIRECTION
       CMPB #3 ; first byte = 3 means following byte is for up/down servo
       BEQ UP_DOWN
       CMPB #4 ; first byte = 4 means following byte is for left/right servo
       BEQ LEFT_RIGHT
CMPB #5 ; first byte = 5 means send sensor readings  
LBEQ GET_READINGS  

CMPB #6 ; Turn on camera or toggle Mode on camera  
LBEQ TURN_ONCAM  

CMPB #7  
LBEQ CAPTURE_PIC  

BRA LOOP

LEFT_MOTOR  
LDAA SC0SR1 ; check status reg (RDRF in bit 5)  
ANDA #%00100000 ; check if receive buffer full  
BEQ LEFT_MOTOR ; wait until data present  

LDAB SC0DRL  
LDA A #0  
LDY #400  
EMUL ; multiply command by 400 and add 1  
ADDD #1  
STD TC0  

BRA LOOP

RIGHT_MOTOR  
LDAA SC0SR1 ; check status reg (RDRF in bit 5)  
ANDA #%00100000 ; check if receive buffer full  
BEQ RIGHT_MOTOR ; wait until data present  

LDAB SC0DRL  
LDA A #0  
LDY #400  
EMUL  
ADDD #1  
STD TC1  

BRA LOOP

DIRECTION  
LDAA SC0SR1 ; check status reg (RDRF in bit 5)  
ANDA #%00100000 ; check if receive buffer full  
BEQ DIRECTION ; wait until data present  

LDAB SC0DRL  
BEQ FORWARD  

movb #%0001010, PORTJ  
BRA LOOP

FORWARD  
movb #%00001010, PORTJ  
LBRA LOOP

UP_DOWN  
LDAA SC0SR1 ; check status reg (RDRF in bit 5)  
ANDA #%00100000 ; check if receive buffer full  
BEQ UP_DOWN ; wait until data present
LDAB SC0DRL
LDA A #0
LDY #14 ; multiply by constant
EMUL
ADDD #1400
STD TC2

LBRA LOOP

LEFT_RIGHT LDA A SC0SR1 ; check status reg (RDRF in bit 5)
ANDA #%00100000 ; check if receive buffer full
BEQ LEFT_RIGHT ; wait until data present
LDA A SC0DRL
LDA A #0
LDY #14 ; multiply by constant
EMUL
ADDD #1400
STD TC3

LBRA LOOP

GET_READINGS LDA A ADR0H ; send IR right
JSR SEND_BYTE
LDA A ADR1H ; send IR left
JSR SEND_BYTE
LDA A PORTG ; send bump sensors
JSR SEND_BYTE
LDA A PORTJ ; send pyro
JSR SEND_BYTE

LBRA LOOP

TURN_ONCAM movb #%00000010, PORTA
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
movb #%00000000, PORTA
LBRA LOOP

CAPTURE_PIC movb #00000001, PORTA
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
JSR WAIT_HALF
movb #00000000, PORTA

LBRA LOOP

*---------------------------------------------------------------
* SUBROUTINES
*---------------------------------------------------------------

*---------------------------------------------------------------
** Send whatever is in accumulator A out the serial port
** when it becomes available.
*---------------------------------------------------------------

SEND_BYTE BRCLR SC0SR1, %10000000, SEND_BYTE
STAA SC0DRL
RTS

************************************************
* Prints out to SCI until EOF (0) is reached by where X is pointing
************************************************

PRINT_SCI BRCLR SC0SR1, BIT7, PRINT_SCI ; Wait for TDR to be empty
MOV B 1, X+, SC0DRL ; Send out char and inc X
TST 0, X
BNE PRINT_SCI ; If not end of string,
continue
RTS ; Otherwise, end subroutine

************************************************
* Prints single char in REG B to SCI
PRINT_CHAR TST SC0SR1 ;wait for transmit data register empty (TDRE)
BPL PRINT_CHAR ; Wait for TDR to be empty
STAB SC0DRL ; Send Out
RTS ; End subroutine

***********************************************************************
** DELAY ROUTINE
** REGISTERS: Y Reg
** 5ms = 12(0.5us) + 6(0.5us)Y - 2(0.5us)
***********************************************************************

DELAY PSHY ; Y REG IS USED, SO SAVING FIRST
LDY #200 ; LOADING Y WITH TIME DELAY
IN_DELAY NOP
NOP
DEY
BNE IN_DELAY
PULY ; PULLING Y TO WHAT IT WAS ORIGINALLY
RTS

*-----------------------------------------------------------
** DELAY ROUTINE
** REGISTERS: Y Reg
** 500us = 12(0.125us) + 6(0.125us)Y - 2(0.125us)
*-----------------------------------------------------------

WAIT_HALF PSHY ; Y REG IS USED, SO SAVING FIRST
LDY #$FFFF ; LOADING Y WITH TIME DELAY
IN_HALF NOP
NOP
DEY
BNE IN_HALF
PULY ; PULLING Y TO WHAT IT WAS ORIGINALLY
RTS

*-----------------------------------------------------------
** DELAY ROUTINE
** REGISTERS: Y Reg
** 100us = 12(0.125us) + 6(0.125us)Y - 2(0.125us)
*-----------------------------------------------------------

WAIT_AD PSHY ; Y REG IS USED, SO SAVING FIRST
LDY #132 ; LOADING Y WITH TIME DELAY
IN_WAIT_AD NOP
NOP
DEY
BNE IN_WAIT_AD
PULY ; PULLING Y TO WHAT IT WAS ORIGINALLY
RTS

***********************************************************************
* INITIALIZATION ROUTINES
***********************************************************************
*---------------------------*  
* INIT SCI SYSTEM *  
*---------------------------*  

INIT_SCI movb #0,SC0CR2 ;disable SCI 0 rcvr. & xmtr. & rx int  
movb #0,SC0CR1  
movb #BAUD9600, SC0BDL ;Set baud rate to 9600  
LDA #%00001100 ; Enable Tx and Rx;  
STAA SC0CR2 ; all interrupts disabled  
ldaa SC0SR1 ; read register to clear flag RDRF  
ldaa SC0DRL ;dummy read to flush receive buffer  
******************End SCI Init*******************  

*---------------------------*  
* INIT MOTORS (only on OCO and OC1) *  
*---------------------------*  

INIT_MOTOR BSET tios,%10000011 ; Setup channels 0,1, and 7 to be TOC channels.  
; Set channels 0 and 1 to clear their output pins when the compare happens  
BSET tctl2, %00001010  
; Set the prescalar to roll over on 32ms periods  
BSET tmsk2,%00110010  
; Setup TOC7 to handle the start of the pulses by setting the value to 1  
; when the TCNT is zero  
BSET oc7m, %00000011  
BSET oc7d, %00000011  
LDX #$9C40  
STX TC7  
BSET tmsk2, %00001000  
; Turn on the timer  
BSET tscr, %00000000  
;Initialize PORTJ  
movb #%00001111,DDRJ ;Bits 0 - 3 of PORTJ are Output  
MOVW #1, tc0  
MOVW #1, tc1  
RTS ;END SERVICE ROUTINE  
******************End Motor Init*******************  

*---------------------------*  
* INIT SERVOS (only on OC2 and OC3) *  
*---------------------------*
INIT_SERVOS  BSET  tios, %10001100 ; Setup channels 2, 3 and 7 to be TOC channels.

; Set channels 2 and 3 to clear their output pins when the compare happens
BSET  tctl2, %10100000

; Set the prescaler to roll over on 32ms periods
BSET  tmsk2, %00110010

; Setup TOC7 to handle the start of the pulses by setting the value to 1 when the TCNT is zero
BSET  oc7m, %00001100
BSET  oc7d, %00001100
LDX  #$9C40
STX  TC7
BSET  tmsk2, %00001000

; Turn on the timer
BSET  tscr, %10000000

MOVW  #1820, tc2
MOVW  #3276, tc3

RTS  ;END SERVICE ROUTINE

***************End Servo Init***************

*------------------------------------------------------------
**  Initialize Analog to Digital converter
*------------------------------------------------------------

INIT_AD  movb  #%10000000, ATDCTL2
jsr  WAIT_AD  ;wait 100us for AD to be ready
movb  #%00110000, ATDCTL5
RTS  ;end subroutine

***************End AD Init***************

*MC68HC812A4 VECTOR INTERRUPTS

VECTOR  EQU  $FFCE
ORG  VECTOR

FDB  EEPROM  ;KEY WAKEUP H
FDB  EEPROM  ;KEY WAKEUP J
FDB  EEPROM  ;ANALOG TO DIGITAL
FDB  EEPROM  ;SERIAL COMMUNICATION 1
FDB  EEPROM  ;SERIAL COMMUNICATION 0
FDB  EEPROM  ;SPI SERIAL TRANSFER COMPLETE
FDB  EEPROM  ;PULSE ACCUMULATOR INPUT EDGE
FDB EEPROM ;PULSE ACCUMULATOR OVERFLOW
FDB EEPROM ;TIMER OVERFLOW
FDB EEPROM ;TIMER CHANNEL 7
FDB EEPROM ;TIMER CHANNEL 6
FDB EEPROM ;TIMER CHANNEL 5
FDB EEPROM ;TIMER CHANNEL 4
FDB EEPROM ;TIMER CHANNEL 3
FDB EEPROM ;TIMER CHANNEL 2
FDB EEPROM ;TIMER CHANNEL 1
FDB EEPROM ;TIMER CHANNEL 0
FDB EEPROM ;REAL TIME INTERRUPT
FDB EEPROM ;IRQ OR KEY WAKE UP D
FDB EEPROM ;XIRQ
FDB EEPROM ;SWI
FDB EEPROM ;RESERVED
FDB EEPROM ;COP FAILURE RESET
FDB EEPROM ;COP CLOCK MONITOR FAIL RESET
FDB EEPROM ;RESET
Appendix C – Pocket PC Code

MAIN MENU CODE

Option Explicit

'GLOBAL VARIABLES
'All forms can access these while program is running

Public leftMotor As Integer
Public rightMotor As Integer
Public Go As Boolean
Public up_down As Integer
Public left_right As Integer
Public IR_right As Integer
Public IR_left As Integer
Public Bump_left As Integer
Public Bump_right As Integer
Public Pyro As Integer
Public Left_Forward As Integer
Public Right_Forward As Integer
Public Left_Reverse As Integer
Public Right_Reverse As Integer
Public leftMotorLeftTurn As Integer
Public rightMotorLeftTurn As Integer
Public leftMotorRightTurn As Integer
Public rightMotorRightTurn As Integer
Public goingInReverse As Boolean

Private Sub ActCalibration_Click()
    Calibration.Show
End Sub

Private Sub ControlPanel_Click()
    Controlpan.Show
End Sub

Private Sub Form_OKClick()
    App.End
End Sub

'Event executed when form loads

Private Sub Form_Load()
    leftMotor = 0    'Initialize all gloabal variables
    rightMotor = 0
    Go = False

up_down = 30  'Servo values are already set when uP boots up
left_right = 134

Left_Forward = 45  'Preset values for motors to make bot go foward
Right_Forward = 80  '    and backwards straight
Left_Reverse = 50
Right_Reverse = 70

leftMotorLeftTurn = 0
rightMotorLeftTurn = 90
leftMotorRightTurn = 90
rightMotorRightTurn = 0

goingInReverse = False

End Sub

Private Sub TakePics_Click()
    Capture.Show
End Sub

Private Sub TestCam_Click()
    Camera.Show
End Sub

CONTROL PANEL CODE

' Desc:  Control program for Little Homie robot
' Date:  11/6/02
' Programer: Jesse Martin

Option Explicit

'Form VARIABLES

Public IncomingStr As String
Public outputBuffer As String
Public OA_on As Boolean

'Event executed when form loads

Private Sub Form_Load()
    Timer1.Enabled = True  'Turn on timer event that gets
                        'sensor readings every .3 sec
    OA_on = False
    Splash.goingInReverse = False
    Comm1.PortOpen = True  'Open Com 1
Private Sub Form_OKClick()
    Timer1.Enabled = False
    Comm1.PortOpen = False
    Controlpan.Hide    'Close Form
End Sub

Private Sub Comm1_OnComm()
    Select Case Comm1.CommEvent
        Case comEvReceive
            IncomingStr = Comm1.Input
            Process_data(IncomingStr)  'Process data when data is received
        Case comEvSend
            ' do nothing here
    End Select
End Sub

Private Sub Process_data(temp As String)
    Dim Size As Integer
    Size = Len(temp)
    If Size = 4 Then    'Only process data when data is 3 bytes long
        IR_rightText.text = CByte(Asc(Mid(temp, 1, 1)))
        Splash.IR_right = CInt(Asc(Mid(temp, 1, 1)))
        IR_leftText.text = CByte(Asc(Mid(temp, 2, 1)))
        Splash.IR_left = CInt(Asc(Mid(temp, 2, 1)))
        Bump_rightText.text = CByte(Asc(Mid(temp, 3, 1))) And 1   'AND with 1 to get value of Bit0
        Splash.Bump_right = CInt(Asc(Mid(temp, 3, 1))) And 1
        If (CByte(Asc(Mid(temp, 3, 1))) And 2) = 2 Then     'AND with 2 to get value of Bit1
            Bump_leftText.text = 1
            Splash.Bump_left = 1
        Else
            Bump_leftText.text = 0
            Splash.Bump_left = 0
        End If
        If (CByte(Asc(Mid(temp, 4, 1))) And 128) = 128 Then    'AND with 128 to get value of Bit7
            Splash.Pyro = 1
            PyroText.text = 1
        Else
            Splash.Pyro = 0
            PyroText.text = 0
        End If
    End If
End Sub

Private Sub OA_off_Click()
    OA_on = False
    Suspend.Caption = "Motors Suspended"
    motorStop_Click
    Splash.leftMotor = 0
    Splash.rightMotor = 0
motorLeftText.text = 0
motorRightText.text = 0
motorForward.Value = True
motorForward_Click

End Sub

Private Sub OA_on_Click()
    OA_on = True
    Suspend.Caption = "OA running"
End Sub

'**********SERVO CONTROLS*************

Private Sub servocenter_Click()
    Splash.left_right = 134
    Text2.text = (Splash.left_right * 14) + 1400
    Splash.up_down = 30
    Text1.text = (Splash.up_down * 14) + 1400
    servogo_Click
End Sub

Private Sub servogo_Click()
    outputBuffer = Chr(4) + Chr(Splash.left_right)
    Comm1.Output = outputBuffer   'Output left/right commands to uP
    outputBuffer = Chr(3) + Chr(Splash.up_down)
    Comm1.Output = outputBuffer   'Output up/down commands to uP
End Sub

Private Sub servoleftlimit_Click()
    Splash.left_right = 255
    Text2.text = (Splash.left_right * 14) + 1400
    Splash.up_down = 11
    Text1.text = (Splash.up_down * 14) + 1400
    servogo_Click
End Sub

Private Sub servoright_Click()
    If (Not Splash.left_right = 255) Then
        Splash.left_right = Splash.left_right + 1
        Text2.text = (Splash.left_right * 14) + 1400
    End If
End Sub

Private Sub servoleft_Click()
    If (Not Splash.left_right = 0) Then
        Splash.left_right = Splash.left_right - 1
        Text2.text = (Splash.left_right * 14) + 1400
    End If
End Sub

Private Sub servorightlimit_Click()
Splash.left_right = 23
Text2.text = (Splash.left_right * 14) + 1400
Splash.up_down = 129
Text1.text = (Splash.up_down * 14) + 1400
servogo_Click
End Sub

Private Sub servoup_Click()
If (Not Splash.up_down = 255) Then
Splash.up_down = Splash.up_down + 1
Text1.text = (Splash.up_down * 14) + 1400
End If
End Sub

Private Sub servodown_Click()
If (Not Splash.up_down = 0) Then
Splash.up_down = Splash.up_down - 1
Text1.text = (Splash.up_down * 14) + 1400
End If
End Sub

'**********END SERVO CONTROLS***********

'Timer event that sends command to get new
'readings from sensors every .3sec

Private Sub Timer1_Timer()
outputBuffer = Chr(5)
Comm1.Output = outputBuffer
If OA_on = True Then
Timer1.Enabled = False  'Must turn off timer until event is know
If Splash.Bump_right = 0 Or Splash.Bump_left = 0 Then
If Splash.goingInReverse = False Then
motorStop_Click   'stop motors
delay_halfsec
motorReverse_Click   'set motors into reverse
Splash.goingInReverse = True
End If
motorReverse_Click
Splash.leftMotor = Splash.Left_Reverse
Splash.rightMotor = Splash.Right_Reverse
motorGo_Click
delay_1sec
If Splash.IR_left > Splash.IR_right Then
If Splash.goingInReverse = True Then
motorStop_Click
delay_halfsec
motorForward_Click   'make sure motors are set to forward
Splash.goingInReverse = False
End If
motorForward_Click
Splash.rightMotor = Splash.rightMotorLeftTurn
Splash.leftMotor = Splash.leftMotorLeftTurn
motorGo_Click
delay_1sec
motorStop_Click

Else
    If Splash.goingInReverse = True Then
        motorStop_Click
delay_halfsec
        motorForward_Click \make sure motors are set to forward
        Splash.goingInReverse = False
    End If
motorForward_Click
Splash.rightMotor = Splash.rightMotorRightTurn
Splash.leftMotor = Splash.leftMotorRightTurn
motorGo_Click
delay_1sec
motorStop_Click
End If

ElseIf Splash.IR_left <= 25 And Splash.IR_right <= 25 Then
    If Splash.goingInReverse = True Then
        motorStop_Click
delay_halfsec
        motorForward_Click \make sure motors are set to forward
        Splash.goingInReverse = False
    End If
motorForward_Click
Splash.leftMotor = Splash.Left_Forward
Splash.rightMotor = Splash.Right_Forward
motorGo_Click \go forward
ElseIf Splash.IR_left > 25 And Splash.IR_right <= 25 Then \there is something to the right
    If Splash.goingInReverse = True Then
        motorStop_Click
delay_halfsec
        motorForward_Click \make sure motors are set to forward
        Splash.goingInReverse = False
    End If
motorForward_Click
Splash.leftMotor = Splash.leftMotorLeftTurn
Splash.rightMotor = Splash.rightMotorLeftTurn
motorGo_Click \turn!
ElseIf Splash.IR_left <= 25 And Splash.IR_right > 25 Then \there is something to the left
    If Splash.goingInReverse = True Then
        motorStop_Click
delay_halfsec
        motorForward_Click \make sure motors are set to forward
        Splash.goingInReverse = False
    End If
motorForward_Click
Splash.leftMotor = Splash.leftMotorRightTurn
Splash.rightMotor = Splash.rightMotorRightTurn
motorGo_Click \turn!
ElseIf Splash.IR_left > 25 And Splash.IR_right > 25 Then
 If Splash.goingInReverse = False Then
 motorStop_Click 'stop motors
delay_halfsec
 motorReverse_Click 'set motors into reverse
 Splash.goingInReverse = True
 End If
 motorReverse_Click
 Splash.leftMotor = Splash.Left_Reverse
 Splash.rightMotor = Splash.Right_Reverse
 motorGo_Click
delay_halfsec 'do nothing for .5 sec

 'get an update
 outputBuffer = Chr(5)
 Comm1.Output = outputBuffer

 If Splash.IR_left <= 25 And Splash.IR_right <= 25 Then
 If Splash.IR_left > Splash.IR_right Then
 If Splash.goingInReverse = True Then
 motorStop_Click
delay_halfsec
 motorForward_Click 'make sure motors are set to forward
 Splash.goingInReverse = False
 End If
 motorForward_Click
 Splash.rightMotor = Splash.rightMotorLeftTurn
 Splash.leftMotor = Splash.leftMotorLeftTurn
 motorGo_Click
delay_1sec
 motorStop_Click

 Else
 If Splash.goingInReverse = True Then
 motorStop_Click
delay_halfsec
 motorForward_Click 'make sure motors are set to forward
 Splash.goingInReverse = False
 End If
 motorForward_Click
 Splash.rightMotor = Splash.rightMotorRightTurn
 Splash.leftMotor = Splash.leftMotorRightTurn
 motorGo_Click
delay_1sec
 motorStop_Click

 End If
 End If
 End If

 Timer1.Enabled = True 'Turn timer back on
 End Sub
Private Sub delay_halfsec()
    Dim Start, Finish As Double
    Start = Timer
    Finish = Start + 0.5
    Do While Timer < Finish
        'Can do other processing, but instead just eating up time
            Loop
    End Sub
End Sub

Private Sub delay_1sec()
    Dim Start, Finish As Double
    Start = Timer
    Finish = Start + 1#
    Do While Timer < Finish
        'Can do other processing, but instead just eating up time
            Loop
    End Sub
End Sub

'**********MOTOR CONTROLS************

Private Sub motorForward_Click()
    outputBuffer = Chr(2) + Chr(0)
    Comm1.Output = outputBuffer
End Sub

Private Sub motorReverse_Click()
    outputBuffer = Chr(2) + Chr(1)
    Comm1.Output = outputBuffer
End Sub

Private Sub motorGo_Click()
    outputBuffer = Chr(0) + Chr(Splash.leftMotor)
    Comm1.Output = outputBuffer
    outputBuffer = Chr(1) + Chr(Splash.rightMotor)
    Comm1.Output = outputBuffer
    Splash.Go = True
    If OA_on = False Then
        Suspend.Caption = ""
    End If
End Sub

Private Sub motorLeftDown_Click()
    If (Not Splash.leftMotor = 0) Then
        Splash.leftMotor = Splash.leftMotor - 1
        motorLeftText.text = CByte(Splash.leftMotor)
        If Splash.Go = True Then
            outputBuffer = Chr(0) + Chr(Splash.leftMotor)
            Comm1.Output = outputBuffer
        End If
    End If
End If
End Sub

Private Sub motorLeftUp_Click()
    If (Not Splash.leftMotor = 100) Then
        Splash.leftMotor = Splash.leftMotor + 1
        motorLeftText.text = CByte(Splash.leftMotor)
        If Splash.Go = True Then
            outputBuffer = Chr(0) + Chr(Splash.leftMotor)
            Comm1.Output = outputBuffer
        End If
    End If
End Sub

Private Sub motorRightDown_Click()
    If (Not Splash.rightMotor = 0) Then
        Splash.rightMotor = Splash.rightMotor - 1
        motorRightText.text = CByte(Splash.rightMotor)
        If Splash.Go = True Then
            outputBuffer = Chr(1) + Chr(Splash.rightMotor)
            Comm1.Output = outputBuffer
        End If
    End If
End Sub

Private Sub motorRightUp_Click()
    If (Not Splash.rightMotor = 100) Then
        Splash.rightMotor = Splash.rightMotor + 1
        motorRightText.text = CByte(Splash.rightMotor)
        If Splash.Go = True Then
            outputBuffer = Chr(1) + Chr(Splash.rightMotor)
            Comm1.Output = outputBuffer
        End If
    End If
End Sub

Private Sub motorStop_Click()
    outputBuffer = Chr(1) + Chr(0)
    Comm1.Output = outputBuffer
    outputBuffer = Chr(0) + Chr(0)
    Comm1.Output = outputBuffer
    Splash.Go = False
    Suspend.Caption = "Motors Suspended"
End Sub

Private Sub motorHalf_Click()
    motorLeftText.text = "50"
    motorRightText.text = "50"
    Splash.leftMotor = 50
    Splash.rightMotor = 50
    If Splash.Go = True Then
        outputBuffer = Chr(0) + Chr(Splash.leftMotor) + Chr(1) + Chr(Splash.rightMotor)
Comm1.Output = outputBuffer
End If
End Sub

'**********END MOTOR CONTROLS**********

ACTUATION CALIBRATION CODE

Option Explicit

Private Sub Form_Load()
    'get values and display in appropriate fields
    leftMotorForward.text = Splash.Left_Forward
    rightMotorForward.text = Splash.Right_Forward
    leftMotorBackwards.text = Splash.Left.Reverse
    rightMotorBackwards.text = Splash.Right.Reverse
    leftMotorLeftTurn.text = Splash.leftMotorLeftTurn
    rightMotorLeftTurn.text = Splash.rightMotorLeftTurn
    leftMotorRightTurn.text = Splash.leftMotorRightTurn
    rightMotorRightTurn.text = Splash.rightMotorRightTurn
End Sub

Private Sub Apply_Click()
    'get changes and store
    Splash.Left.Forward = leftMotorForward.text
    Splash.Right.Forward = rightMotorForward.text
    Splash.Left.Reverse = leftMotorBackwards.text
    Splash.Right.Reverse = rightMotorBackwards.text
    Splash.leftMotorLeftTurn = leftMotorLeftTurn.text
    Splash.rightMotorLeftTurn = rightMotorLeftTurn.text
    Splash.leftMotorRightTurn = leftMotorRightTurn.text
    Splash.rightMotorRightTurn = rightMotorRightTurn.text
End Sub

Private Sub Close_Click()
    Calibration.Hide 'Close Form
End Sub

Private Sub Form_OKClick()
    Calibration.Hide 'Close Form
End Sub

TEST CAMERA CODE

Option Explicit

Private outputBuffer As String

Private Sub Form_Load()
    Comm1.PortOpen = True 'Open Com 1
End Sub
Private Sub Form_OKClick()
    Comm1.PortOpen = False
    Camera.Hide
End Sub

Private Sub Erase_Pic_Click()
    outputBuffer = Chr(6)           'Toggle through Modes
    Comm1.Output = outputBuffer
    outputBuffer = Chr(6)
    Comm1.Output = outputBuffer
    outputBuffer = Chr(6)
    Comm1.Output = outputBuffer
    outputBuffer = Chr(6)
    Comm1.Output = outputBuffer
    delay_2sec
    outputBuffer = Chr(7)           'Confirm erase
    Comm1.Output = outputBuffer
    outputBuffer = Chr(7)
    Comm1.Output = outputBuffer
End Sub

Private Sub Take_Pic_Click()
    outputBuffer = Chr(7)
    Comm1.Output = outputBuffer
End Sub

Private Sub Turn_on_Click()
    outputBuffer = Chr(6)
    Comm1.Output = outputBuffer
End Sub

Private Sub delay_2sec()
    Dim Start, Finish As Double
    Start = Timer
    Finish = Start + 2#
    Do While Timer < Finish
        'Can do other processing, but instead just eating up time
    Loop
End Sub

**TAKE PICTURES CODE**

Option Explicit

Private Start_on As Boolean         'Robot's main behavior is running
Private IncomingStr As String
Private outputBuffer As String
Private pyro_found As Boolean
Private wait_mode As Boolean       'In wait mode to take pic?
Private pictureTaken As Boolean
Private pictureNumber As Integer
Private Count As Integer

Public text As String
Public temp As Long
Public Const SND_SYNC = &H0 ' play synchronously (default)
Public Const SND_ASYNC = &H1 ' play asynchronously
Public Declare Function PlaySound Lib "Coredll" Alias "PlaySoundW" (ByVal lpzName As String, ByVal hModule As Long, ByVal dwFlags As Long) As Long

Private Sub Form_Load()
    Comm1.PortOpen = True
    Timer1.Enabled = True
    Information.text = "(" & Time & ") - Waiting for pyro to stabilize..." & (Chr(10)) & Information.text
    wait_mode = False
    Splash.goingInReverse = False
    pictureNumber = 1
    Count = 1
End Sub

Private Sub Form_OKClick()
    Timer1.Enabled = False
    Timer2.Enabled = False
    Timer3.Enabled = False
    Timer4.Enabled = False
    Timer5.Enabled = False
    StopButton_Click
    Capture.Hide
    Comm1.PortOpen = False
End Sub

Private Sub Find_Pyro()
    'Found first pyro reading?
    If Splash.Pyro = 1 Then
        Timer2.Enabled = False
        Information.text = "(" & Time & ") - Pyro ready; push start to begin..." & (Chr(10)) & Information.text
        Start.Enabled = True
    End If
End Sub

Private Sub Start_Click()
    Start_on = True
    Information.text = "(" & Time & ") - Routine Started" & (Chr(10)) & Information.text
    Start.Enabled = False
    StopButton.Enabled = True

    'Turn on camera
Turn_on_Click
Timer5.Enabled = True

Timer3.Interval = Int((30000 - 5000 + 1) * Rnd + 5000)    'generating time between 5sec - 30sec
Timer3.Enabled = True

End Sub

Private Sub StopButton_Click()
    Start_on = False
    Information.text = "(" & Time & ") - Routine Suspended" & (Chr(10)) & Information.text
    Start.Enabled = True
    StopButton.Enabled = False
    motorStop_Click   'stop motors

    'Stop all timers
    Timer1.Enabled = False
    Timer2.Enabled = False
    Timer3.Enabled = False
    Timer4.Enabled = False
    Timer5.Enabled = False

End Sub

Private Sub Timer1_Timer()      'refresh sensor readings every 300 ms
    outputBuffer = Chr(5)
    Comm1.Output = outputBuffer
    If Start_on = True Then
        Timer1.Enabled = False  'Must turn off timer until event is know
        If Splash.Bump_right = 0 Or Splash.Bump_left = 0 Then
            If Splash.goingInReverse = False Then
                motorStop_Click   'stop motors
delay_halfsec
                motorReverse_Click   'set motors into reverse
                Splash.goingInReverse = True
            End If
        End If
        motorReverse_Click
        Splash.leftMotor = Splash.Left_Reverse
        Splash.rightMotor = Splash.Right_Reverse
        motorGo_Click
delay_1sec
    If Splash.IR_left > Splash.IR_right Then
        If Splash.goingInReverse = True Then
            motorStop_Click
delay_halfsec
            motorForward_Click   'make sure motors are set to forward
            Splash.goingInReverse = False
        End If
    End If
    motorForward_Click
    Splash.rightMotor = Splash.rightMotorLeftTurn
    Splash.leftMotor = Splash.leftMotorLeftTurn
    motorGo_Click
delay_1sec
motorStop_Click

Else
    If Splash.goingInReverse = True Then
        motorStop_Click
        delay_halfsec
        motorForward_Click  'make sure motors are set to forward
        Splash.goingInReverse = False
    End If
    motorForward_Click
    Splash.rightMotor = Splash.rightMotorRightTurn
    Splash.leftMotor = Splash.leftMotorRightTurn
    motorGo_Click
    delay_1sec
    motorStop_Click
End If
ElseIf Splash.IR_left <= 25 And Splash.IR_right <= 25 Then
    If Splash.goingInReverse = True Then
        motorStop_Click
        delay_halfsec
        motorForward_Click  'make sure motors are set to forward
        Splash.goingInReverse = False
    End If
    motorForward_Click
    Splash.leftMotor = Splash.Left_Forward
    Splash.rightMotor = Splash.rightMotorLeftTurn
    motorGo_Click    'turn!
ElseIf Splash.IR_left > 25 And Splash.IR_right <= 25 Then   'there is something to the right
    If Splash.goingInReverse = True Then
        motorStop_Click
        delay_halfsec
        motorForward_Click  'make sure motors are set to forward
        Splash.goingInReverse = False
    End If
    motorForward_Click
    Splash.leftMotor = Splash.leftMotorRightTurn
    Splash.rightMotor = Splash.rightMotorRightTurn
    motorGo_Click   'turn!
ElseIf Splash.IR_left <= 25 And Splash.IR_right > 25 Then   'there is something to the left
    If Splash.goingInReverse = True Then
        motorStop_Click
        delay_halfsec
        motorForward_Click  'make sure motors are set to forward
        Splash.goingInReverse = False
    End If
    motorForward_Click
    Splash.leftMotor = Splash.leftMotorRightTurn
    Splash.rightMotor = Splash.rightMotorRightTurn
    motorGo_Click   'turn!
ElseIf Splash.IR_left > 25 And Splash.IR_right > 25 Then
    If Splash.goingInReverse = False Then
        motorStop_Click  'stop motors
delay_halfsec
motorReverse_Click 'set motors into reverse
Splash.goingInReverse = True
End If
motorReverse_Click
Splash.leftMotor = Splash.Left_Reverse
Splash.rightMotor = Splash.Right_Reverse
motorGo_Click
delay_halfsec 'do nothing for .5 sec

'get an update
outputBuffer = Chr(5)
Comm1.Output = outputBuffer

If Splash.IR_left <= 25 And Splash.IR_right <= 25 Then ' 
  If Splash.IR_left > Splash.IR_right Then
    If Splash.goingInReverse = True Then
      motorStop_Click
delay_halfsec
      motorForward_Click 'make sure motors are set to forward
      Splash.goingInReverse = False
    End If
  End If
  motorForward_Click
  Splash.rightMotor = Splash.rightMotorLeftTurn
  Splash.leftMotor = Splash.leftMotorLeftTurn
  motorGo_Click
delay_1sec
  motorStop_Click
Else
  If Splash.goingInReverse = True Then
    motorStop_Click
delay_halfsec
    motorForward_Click 'make sure motors are set to forward
    Splash.goingInReverse = False
  End If
  motorForward_Click
  Splash.rightMotor = Splash.rightMotorRightTurn
  Splash.leftMotor = Splash.leftMotorRightTurn
  motorGo_Click
delay_1sec
  motorStop_Click
End If
End If
ElseIf wait_mode = True And Start_on = False Then
  If Splash.Pyro = 1 Then 'If person detected, then take pic
    Take_Pic_Click
    Information.text = "(" & Time & ") - Picture " & pictureNumber & " taken..." & (Chr(10)) & Information.text

  'Stop taking pictures
Timer4_Timer
End If
End If

Timer1.Enabled = True  'Turn timer back on
End Sub

Private Sub Comm1_OnComm()
Select Case Comm1.CommEvent
Case comEvReceive
    IncomingStr = Comm1.Input
    Process_data(IncomingStr)  'Process data when data is received
Case comEvSend
    'do nothing here
End Select
End Sub

Private Sub Process_data(temp As String)
    Dim Size As Integer
    Size = Len(temp)
    If Size = 4 Then  'Only process data when data is 3 bytes long
        Splash.IR_right = CInt(Asc(Mid(temp, 1, 1)))
        Splash.IR_left = CInt(Asc(Mid(temp, 2, 1)))
        Splash.Bump_right = CInt(Asc(Mid(temp, 3, 1))) And 1
        If (CByte(Asc(Mid(temp, 3, 1))) And 2) = 2 Then  'AND with 2 to get value of Bit1
            Splash.Bump_left = 1
        Else
            Splash.Bump_left = 0
        End If
        If (CByte(Asc(Mid(temp, 4, 1))) And 128) = 128 Then  'AND with 128 to get value of Bit7
            Splash.Pyro = 1
        Else
            Splash.Pyro = 0
        End If
    End If
End Sub

Private Sub delay_halfsec()
    Dim Start, Finish As Double
    Start = Timer
    Finish = Start + 0.5
    Do While Timer < Finish  
        'Can do other processing, but instead just eating up time
        Loop
    End Sub

Private Sub delay_1sec()
    Dim Start, Finish As Double
    Start = Timer
    Finish = Start + 1#
    Do While Timer < Finish  
        'Can do other processing, but instead just eating up time
    Loop
Loop
End Sub

'**********MOTOR CONTROLS**********

Private Sub motorForward_Click()
    outputBuffer = Chr(2) + Chr(0)
    Comm1.Output = outputBuffer
    Splash.goingInReverse = False
End Sub

Private Sub motorReverse_Click()
    outputBuffer = Chr(2) + Chr(1)
    Comm1.Output = outputBuffer
    Splash.goingInReverse = True
End Sub

Private Sub motorGo_Click()
    outputBuffer = Chr(0) + Chr(Splash.leftMotor)
    Comm1.Output = outputBuffer
    outputBuffer = Chr(1) + Chr(Splash.rightMotor)
    Comm1.Output = outputBuffer
    Splash.Go = True
End Sub

Private Sub motorStop_Click()
    outputBuffer = Chr(1) + Chr(0)
    Comm1.Output = outputBuffer
    outputBuffer = Chr(0) + Chr(0)
    Comm1.Output = outputBuffer
    Splash.Go = False
End Sub

'**********END MOTOR CONTROLS**********

Private Sub servogo_Click()
    outputBuffer = Chr(4) + Chr(Splash.left_right)
    Comm1.Output = outputBuffer  'Output left/right commands to uP
    outputBuffer = Chr(3) + Chr(Splash.up_down)
    Comm1.Output = outputBuffer  'Output up/down commands to uP
End Sub

Private Sub Timer2_Timer()
    Timer2.Interval = 300  'make sure interval is at 300 ms and see if
                         'pyro is ready
    Find_Pyro
End Sub

Private Sub Timer3_Timer()
Timer3.Enabled = False  'Turn off timer that caused this event
Start_on = False        'Stop OA

temp = PlaySound("abouttotake" & Count Mod 4 & ".wav", 0, SND_ASYNC)
'stop motors
motorStop_Click

'Reandomly generate up/down and left/right position on pan-and-tilt
Splash.up_down = Int((129 - 40 + 1) * Rnd + 40)
Splash.left_right = Int((255 - 23 + 1) * Rnd + 23)
servogo_Click
delay_halfsec

Timer4.Enabled = True    'Turn on 5 sec timer

End Sub

Private Sub Take_Pic_Click()
   outputBuffer = Chr(7)
   Comm1.Output = outputBuffer
   Timer5.Interval = 62000       'reset timer that keeps camera on
   pictureTaken = True
End Sub

Private Sub Turn_on_Click()
   outputBuffer = Chr(6)
   Comm1.Output = outputBuffer
End Sub

Private Sub Keep_on()
   outputBuffer = Chr(6)
   Comm1.Output = outputBuffer
End Sub

Private Sub Timer4_Timer()          'event occurs when time to wait to take pic
   If wait_mode = False Then
      wait_mode = True
      pictureTaken = False
   Else
      Timer4.Enabled = False
   End If

   'generate time when OA will stop again
   Timer3.Interval = Int((30000 - 5000 + 1) * Rnd + 5000)    'generating time between 5sec - 30sec
   Timer3.Enabled = True

   wait_mode = False
   'randomly spin in some direction
   If Splash.IR_left > Splash.IR_right Then
      If Splash.goingInReverse = True Then

motorStop_Click
delay_halfsec
motorForward_Click 'make sure motors are set to forward
Splash.goingInReverse = False
End If
Splash.rightMotor = Splash.rightMotorLeftTurn
Splash.leftMotor = Splash.leftMotorLeftTurn
motorGo_Click
delay_1sec
motorStop_Click

Else
If Splash.goingInReverse = True Then
    motorStop_Click
delay_halfsec
    motorForward_Click 'make sure motors are set to forward
    Splash.goingInReverse = False
End If
Splash.rightMotor = Splash.rightMotorRightTurn
Splash.leftMotor = Splash.leftMotorRightTurn
motorGo_Click
delay_1sec
motorStop_Click
End If

If pictureTaken = True Then
    temp = PlaySound("picturetaken" & Count Mod 5 & ".wav", 0, SND_ASYNC)
pictureNumber = pictureNumber + 1
Count = Count + 1
Else
    temp = PlaySound("nopicturetaken.wav", 0, SND_ASYNC)
    Count = Count + 1
End If

'Go straight forward
Splash.leftMotor = Splash.Left_Foward
Splash.rightMotor = Splash.Right_Foward
motorGo_Click
Start_on = True 'start OA again

End If
End Sub

Private Sub Timer5_Timer() 'Keeps turning on camera
    Keep_on 'Keep Camera On
End Sub