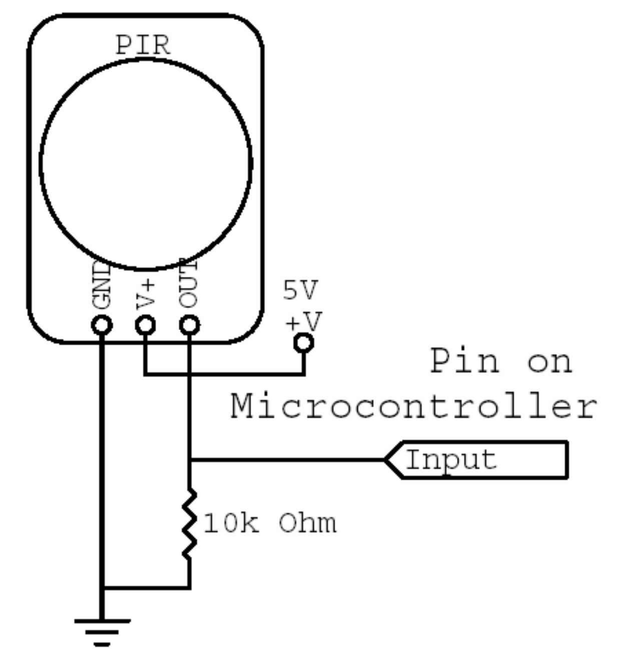
Motion Sensing with the Pyroeletric Sensor

A large amount of time was spent, trying to provide an accurate and reliable way to detect humans. 3 Attempts were made until I finally found a working algorithm and sensor to provide such an accurate and reliable detection. The first and second attempts used a different pyroeletric sensor but used the same detection algorithm with a slight variation. The third attempt used the same sensor as the second attempt but used a different detection algorithm.

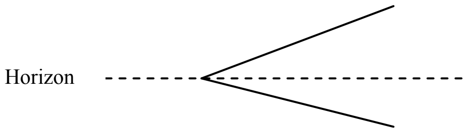
Attempt #	Sensor Used	Algorithm Used
1	1	1 & 1b
2	2	1c
3	2	2

Sensor 1 – HVW Technologies PIR Sensor

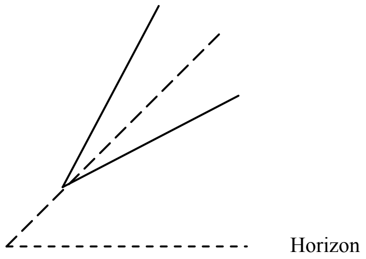


- Passive Infrared Sensor
- Range: 60º Omnidirectional
- Distance: 5m
- Pulses high on signal when motion is detected for approximately 1 second
- $10.50 from hvwtech
- http://www.hvwtech.com/dnload/PIRManual10.pdf

The pir sensor will be mounted 8” above the ground on top of a servo on the serverBOT. To provide a higher angle of detection along the horizon, the pir sensor will be mounted at a 45 degree angle. This elevated angle will set the elevated detection angle from 0-30 degrees to 30-60 degrees from the horizon.

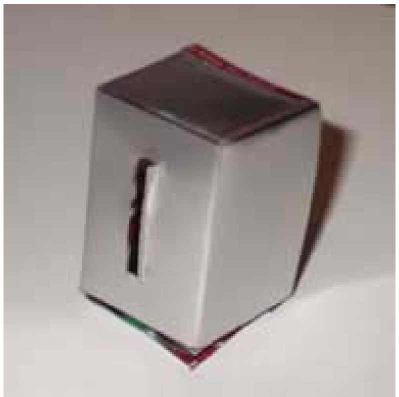


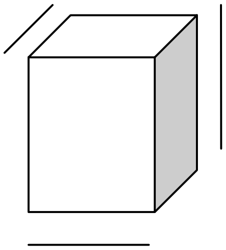
60 degrees



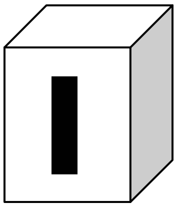
45 degree offset

A cut out aluminum can will be used to limit the active area of the pir sensor to approximately 15 degrees. The device is shaped like a box and fits right over the pir sensor assembly. In the center of this device is a rectangular cutout to limit the azimuth of detection for the pir sensor.

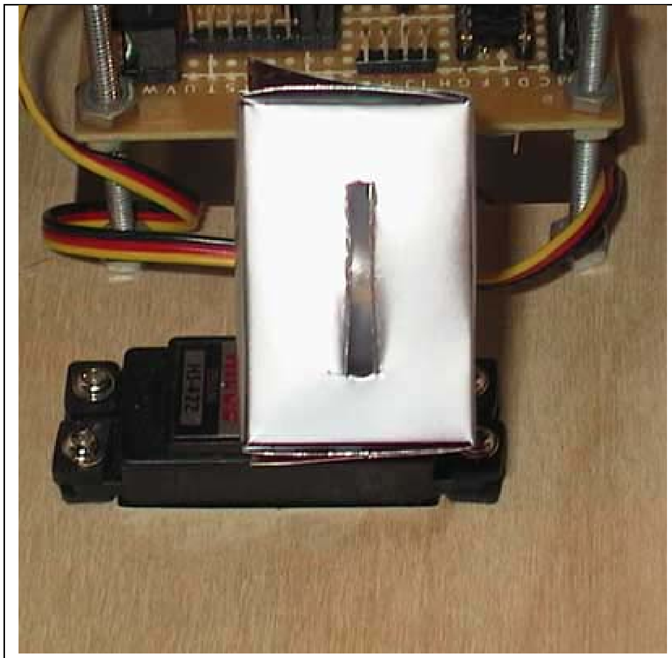
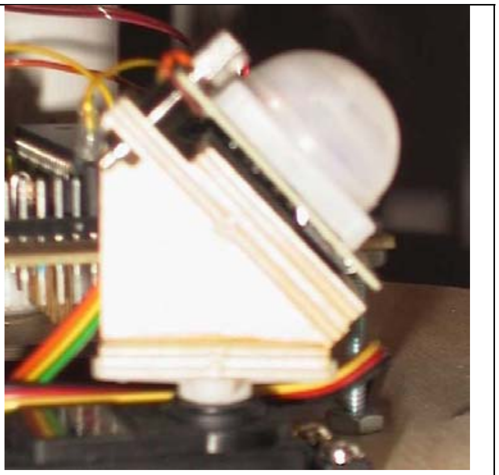




Aluminum shield dimentions:
LxWxH = 1.5”x1”x1”

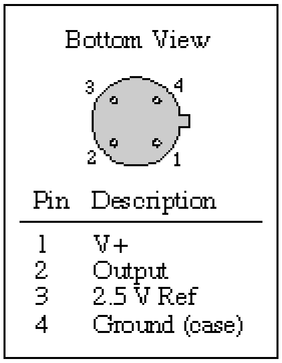


Aluminum shield with cutout hole to limit the azimuth to < 15 degrees.

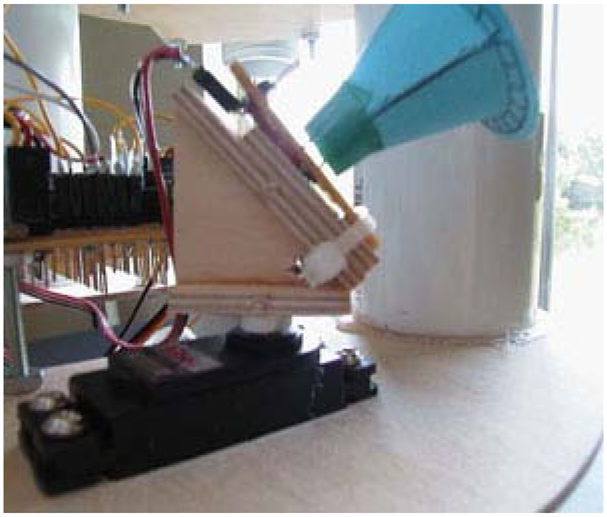
Sensor 2: Eltec 443-2 Pyroelectric Detector

- Analog Output
- More immune to IR noise
- Better Fresnel Lens
- Analog output gives more flexibility and allows the user to set the thresholds of detection properly
- Floats around 2.5V when there is no detection and voltage moves up or down depending on the direction of motion across the dual elements
- http://www.acroname.com/robotics/parts/R1-442-3.html



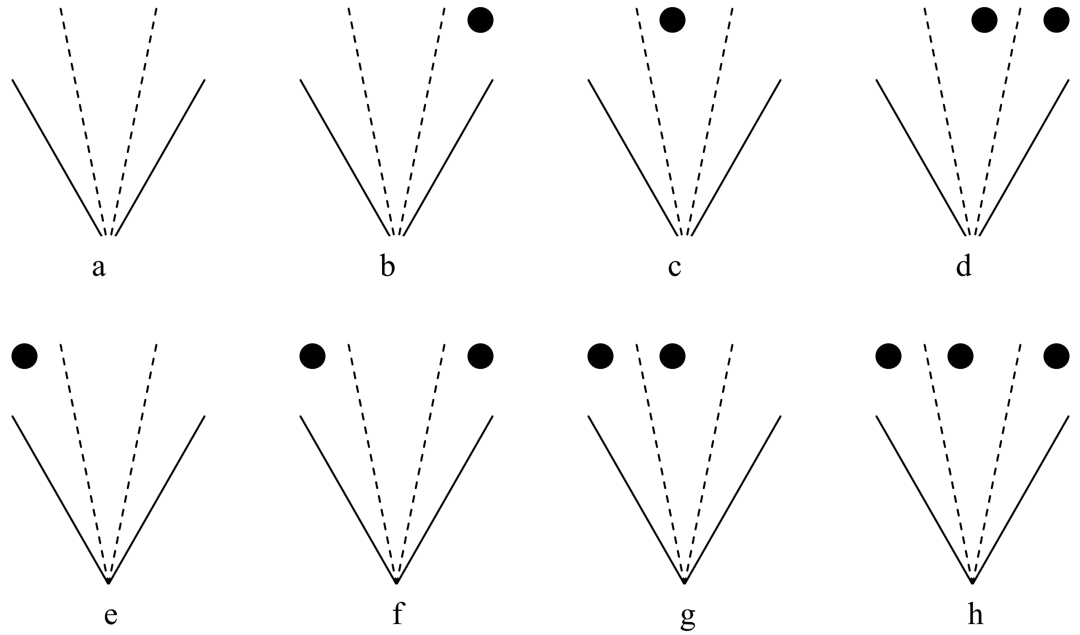
Pin	Description
1	V+
2	Output
3	2.5 V Ref
4	Ground (case)

The eltec 443-2 was mounted the wedge the same way the first sensor was mounted except the fresnel lens provided with the eltec 443-2 was cone shaped.



Sensing Algorithm 1

The basic theory of human detection with a direction vector is to slowly sweeping while pausing in between each increment to see if a human was sensed at that directional vector. The design I created is a slight variation of this basic theory. In my design I will sweep the pir sensor in a constant velocity twice, once in the clockwise direction and once in the counter-clockwise direction. While generating a sweep the uController will record the first detected value in each direction. By using this method in combination with fuzzy logic, the uController can determine which direction the human will be. This also creates a basic algorithm in which there are multiple humans and determines which direction the serverBOT should head. This will provide 8 possible scenarios. In the following chart, the 8 possible scenarios are depicted with the black dot representing a person or a cluster of people.



Using these 8 possible scenarios and the concept of fuzzy logic, the uController will determine which direction to head towards. The range of detection for the pir sensor is from $0E to $21 providing the following boundaries. Using the values on the left of the table below to produce values for the clockwise and counter-clockwise detection. It is placed into the table on the right to provide a fuzzy logic response to determine the direction to head towards.

Left – $0E to $12
Middle – $15 to $1B
Right - $1C to $21

CW:CCW	**Left**	**Middle**	**Right**
Left	**L**	**LM**	**M**
Middle	**LM**	**M**	**RM**
Right	**M**	**RM**	**R**

Sample Code

```
;*****************************************************************************************
;* Subroutine - Motion_Sweep
;* Description: This subroutine is used to sweep the motion sensor CW then CCW
;*                  as the motion detector is being swept, only the first value obtained
;*                  in each direction of motion is recorded resulting in two values
;*                  one stored in DET_CW, and DET_CCW
;* Uses: TEMP
;* Output: DET_CW, DET_CCW
;*****************************************************************************************
Motion_SWEEP:   sbr STATUS, $01         ; Setup STATUS to show CW Sweep
                clr DET_CW              ;
                clr DET_CCW             ;
                rcall TIM2_PWM          ; Set Timer2 to PWM Mode and PortD OC2 to output
                ldi TEMP, $0E           ; Set Servo to Left Most Position
                mov MOT_LOC, TEMP
                out OCR2, MOT_LOC       ;

                rcall Motion_Delay   ; Wait until SERVO SWINGS to leftmost position
                rcall INT0_EN           ; Enable External Interrupt 0

                inc MOT_LOC             ;
SWEEP_LP1:      out OCR2, MOT_LOC       ; Incremental Clockwise Servo Swing

                rcall Motion_Delay
                inc MOT_LOC             ;
                mov TEMP, MOT_LOC
                cpi TEMP, $22           ; Check for Rightmost Position
                brne SWEEP_LP1

                cbr STATUS, $01         ; Setup STATUS to show CCW Sweep
SWEEP_LP2:      out OCR2, MOT_LOC                ;

                rcall Motion_Delay
                dec MOT_LOC
                mov TEMP, MOT_LOC                       ;
                cpi TEMP, $0E               ; Check for Leftmost position
                brne SWEEP_LP2

                rcall INT0_DIS              ; Disable External Interrupt 0

;               ldi TEMP, $18               ; Return to Neutral Position
;               out OCR2, TEMP              ;
;               rcall Motion_Delay  ; Wait until servo swings to neutral position
                ret
;*********************************************************************************
;* Subroutine - External Interrupt 0 Enable
;* Description: Enables the External Interrupt 0 and sets it to trigger off of the
;*              rising edge
;*********************************************************************************
INT0_EN: push TEMP
                ldi TEMP, $03                           ; Set INT0 to trigger off of rising edge
                out MCUCR, TEMP

                ldi TEMP, $40                           ; Enable External Interrupt 0
                out GIMSK, TEMP
                pop TEMP
                ret
;*********************************************************************************
;* Subroutine - External Interrupt 0 Disable
;* Description: Disables the External Interrupt 0
;*********************************************************************************
INT0_DIS:       ldi TEMP, $00                           ; Disable External Interrupt 0
                out GIMSK, TEMP
                ret
;*********************************************************************************
;* Interrupt - External Interrupt 0 Handler
;* Description: When an external interrupt occurs on INT0, a motion is detected by
;*              the sensor. The external Interrupt will check the current direction
;*              and the direction vector to make sure it is the first value for that
;*              direction and if so it records it else it exits
;* Uses: TEMP
;*********************************************************************************
EXT_INT0:       push TEMP                               ;
;               mov GENIOI, TEMP                        ; Debug test, output all values
;               rcall HEX2ASCII                         ; detected and exit interrupt
;               pop TEMP                                ;
;               reti                                    ;

                sbrc STATUS, 0
                rjmp EXINT0_CW
EXTINT0_CCW:    tst DET_CCW                             ; Check to see if First Value of CCW
                brne EXINT0_EX                          ; If not exit
                in TEMP, OCR2
                mov DET_CCW, TEMP                       ; Else record first value on CCW

EXINT0_EX:      pop TEMP                                ;
                reti

EXINT0_CW:      tst DET_CW                              ; Check to see if First value of CW
                brne EXINT0_EX                          ; If not exit
                in TEMP, OCR2
                mov DET_CW, TEMP                        ; Else record first value on CW

                rjmp EXINT0_EX
;*
```

Sensing Algorithm 1b

Algorithm 1b is a variation in algorithm 1 in the sense that when a hit was detected, the sweeping stopped and waited another 4 seconds in the same position of the hit. If another hit was detected than it would count as a human if it wasn't detected within the timeout period, then the servo would continue to sweep the motion sensor.

Sample Code

```
;***************************************************************************************
;* Interrupt - External Interrupt 0 Handler
;* Description: When an external interrupt occurs on INT0, a motion is detected by
;*              the sensor. The external Interrupt will check the current direction
;*              and the direction vector to make sure it is the first value for that
;*              direction and if so it records it else it exits
;* Uses: TEMP
;***************************************************************************************
EXT_INT0:       push TEMP                       ;
;               mov GENIOI, TEMP                ; Debug test, output all values
;               rcall HEX2ASCII                 ; detected and exit interrupt
;               pop TEMP                        ;
;               reti                            ;
                sbrc STATUS, 0                  ; If status bit0 = 0 then execute CCW
                rjmp EXINT0_CW                  ; else execute CW

EXTINT0_CCW:    tst DET_CCW                     ; Check to see if First Value of CCW
                brne EXINT0_EX                  ; If not exit else record first value on CCW

                rcall MOT_ErrAv                 ; Call Motin Detection Error Avoidance
                tst GENIOI                      ; Check GENIOI
                brne EXINT0_EX                  ; If GENIOI != 0 then bad value and exit

                in TEMP, OCR2                   ; record first value on CCW
                mov DET_CCW, TEMP

EXINT0_EX:      pop TEMP                        ;
                reti

EXINT0_CW:      tst DET_CW                      ; Check to see if First value of CW
                brne EXINT0_EX                  ; If not exit else record first value on CW

                rcall MOT_ErrAv                 ; Call Motion Detection Error Avoidance
                tst GENIOI                      ; Check GENIOI
                brne EXINT0_EX                  ; If GENIOI != 0 then bad value and exit

                in TEMP, OCR2                   ; record first value on CW
                mov DET_CW, TEMP

                rjmp EXINT0_EX

                ; Error Avoidance alrogithm
MOT_ErrAv:      rcall INT0_DIS                  ; Disable EXT_INT0 so it cannot be
                                                ; retriggered once global interrupts
                                                ; are re-enabled for nested interrupts

                cbi DDRD, 2                     ; Make sure PORTD bit 2 is set to input
EXINT0_CLR:     sbic PIND, 2                    ; Wait for EXT_INT0: falling edge
                rjmp EXINT0_CLR                 ;

                cbr STATUS, $02                 ; Clear Timeout bit in STATUS
;               andi STATUS, $FD                ;
```

```
                sei                             ; enable global interrupts for TIMER2_INT
                CALL_TIMER1 MOT_Timeout,$02     ; MACRO MOT_Timeout
EXINT0_WT1:     sbrc STATUS, 1                  ; Check if Timer Timeout has occured
                rjmp MOT_ErrAvEX2               ; If so Exit External Interrupt

                sbis PIND, 2                    ; Check for rising edge on ext_int0
                rjmp EXINT0_WT1                 ; Wait for Timeout condition or another
                                                ; trigger on PORTD bit 1
MOT_ErrAvEX:    cli                             ; Disable Global interrutps so it cannot
                                                ; intefere with the rest of this interrupt
                rcall INT0_EN                   ; Re-enable EXT_INT0
                clr GENIOI
                ret                             ; return from subroutine

MOT_ErrAvEX2:   cli
                rcall INT0_EN
                ser GENIOI
                ret
                ;End Error Avoidance Algorithm
```

Sensing Algorithm 1c

Sensing Algorithm 1c is a slight variation of Sensing algorithm 1. Since a analog signal is now used to detect motion the code had to be modified to provide a check on the analog port while swinging. The provides less code because it did not need to use the external interrupts.

Sample Code

```
;*****************************************************************************************
;* Subroutine - Motion_Sweep
;* Description: This subroutine is used to sweep the motion sensor CW then CCW
;*              as the motion detector is being swept, only the first value obtained
;*              in each direction of motion is recorded resulting in two values
;*              one stored in DET_CW, and DET_CCW
;* Uses: TEMP
;* Output: DET_CW, DET_CCW
;*****************************************************************************************
Motion_SWEEP:   rcall Init_ADC

                sbr STATUS, $01                 ; Setup STATUS to show CW Sweep
                clr DET_CW                      ;
                clr DET_CCW                     ;
                rcall TIM2_PWM                  ; Set Timer2 to PWM Mode and PortD OC2 to output
                ldi TEMP, $0E                   ; Set Servo to Left Most Position
                mov MOT_LOC, TEMP
                out OCR2, MOT_LOC               ;

                ldi TEMP2, $08                  ;
                rcall Motion_Delay          ; Wait until SERVO SWINGS to leftmost position

                inc MOT_LOC                     ;
SWEEP_LP1:      out OCR2, MOT_LOC               ; Incremental Clockwise Servo Swing

                ldi TEMP2, $08
                rcall Motion_Delay
                ;
                rcall Motion_Detect         ; Check Motion_Detector
                ;
                inc MOT_LOC                     ;
                mov TEMP, MOT_LOC
                cpi TEMP, $21                   ; Check for Rightmost Position
                brne SWEEP_LP1

                ldi TEMP2, $0C
                rcall Motion_Delay
                dec MOT_LOC
                cbr STATUS, $01                 ; Setup STATUS to show CCW Sweep
SWEEP_LP2:      out OCR2, MOT_LOC               ;

                ldi TEMP2, $08
                rcall Motion_Delay
;
                rcall Motion_Detect     ; Check Motion_Detector
;
                dec MOT_LOC
                mov TEMP, MOT_LOC               ;
                cpi TEMP, $0E                   ; Check for Leftmost position
                brne SWEEP_LP2

                ret
;*
;********************************************************************************
;* Interrupt - External Interrupt 0 Handler
;* Description: When an external interrupt occurs on INT0, a motion is detected by
;*              the sensor. The external Interrupt will check the current direction
;*              and the direction vector to make sure it is the first value for that
;*              direction and if so it records it else it exits
;* Uses: TEMP
;********************************************************************************
Motion_Detect:  push TEMP
                sbrc STATUS, 0                  ; If status bit0 = 0 then execute CCW
                rjmp MOTDET_CW                  ; else execute CW

MOTDET_CCW:     tst DET_CCW                     ; Check to see if First Value of CCW
                brne MOTDET_EX                  ; If not exit else record first value on CCW

                ldi GENIOI, $02                 ;
                rcall ADC_OneRun        ; Call ADC_OneRun with ADMUX set to port2
                cpi GENIOI, $00                 ; $00                   ;
                ;brne MOTDET_EX                 ; If GENIOI != 0 then bad value and exit
                brne CCW_ALT                    ; If GENIOI != 0 then bad value and exit

CCW_REC:        in TEMP, OCR2                   ; record first value on CCW
                mov DET_CCW, TEMP

MOTDET_EX:      pop TEMP                        ;
                reti

CCW_ALT:        cpi GENIOI, $01                 ; Secondary Threshold for CCW = $01 72
                brne MOTDET_EX                  ; if higher byte != $01 then exit
                mov GENIOI, GENIOR              ;
                cpi GENIOI, $AF                 ; Check if lower byte < $72
                brlo CCW_REC                    ;
                rjmp MOTDET_EX                  ; If not exit

MOTDET_CW:      tst DET_CW                      ; Check to see if First value of CW
                brne MOTDET_EX                  ; If not exit else record first value on CW

                ldi GENIOI, $02                 ; Call ADC_OneRun with ADMUX set to port2
                rcall ADC_OneRun        ;
                cpi GENIOI, $03                 ; $03
                ;brne MOTDET_EX                 ; If GENIOI != 0 then bad value and exit
                brne CW_ALT                     ; If GENIOI != 0 then bad value and exit

CW_REC:                 in TEMP, OCR2                   ; record first value on CW
                mov DET_CW, TEMP

                rjmp MOTDET_EX
CW_ALT:         cpi GENIOI, $02                 ; Secondary Threshold for CCW = $02 80
                brne MOTDET_EX                  ; if higher byte != $01 then exit
```

```
mov GENIOI, GENIOR                  ;
cpi GENIOI, $80                     ; Check if lower byte < $80
brlo MOTDET_EX                      ; If so exit
rjmp CW_REC                         ; else record value
```

Sensing Algorithm 2

Algorithm 2 uses status bits are used to indicate a motion detector hit and a direction of motion detection. In one detection cycle, only one clockwise or counter-clockwise sweep is made. The sweep that is made in that detection cycle is determined by the status bit. While the servo is sweeping the motion detector, the motion detector analog values are checked for a human (hit). If there is a hit, then the detection cycle ends and the results are analyzed making a decision to turn or not. If there are no hits detected when the servo reaches the end of the motion sweep, the status bit is toggled so that the next motion sweep goes in the other direction. Example: If the first sweep is CW, and no hits are found then the next week will be a CCW sweep.

Since there is only one sweep, the range fro $0E-$21 on the single sweep is compared to values to generate the 5 degrees of turning resolution.

- Left - $0E-$14
- Middle Left - $15-$17
- Middle - $18-$1B
- Middle Right - $1C-$1E
- Right - $1F-$21

In order to provide a smarter sweeping and detection algorithm, if a hit was made in the middle left or left direction, the status bit was changed so that the sweep starts in the left. If a hit was detected on the middle right or right direction then the sweep would start from the right.

Algorithm 2 also counted the number of No Detects, and when a certain number of No Detect conditions was met, it would randomly turn serverBOT in any direction.

Algorithm 2 is different from algorithm 1 because the motion detector is swept at a faster speed, and swept while the serverBOT is in motion. Algorithm 2 also produces 1 sweep per detection cycle. With algorithm 1, the behavior of serverBOT would check the motion detector and then check the other analog ports in sequential order. In algorithm 2, since the serverBOT is in motion while detecting for a human hit, the other analog ports are checked at the same time the motion detector is checked.

Sample Code

```
;*************************************************************************************
;* Subroutine - Motion_Sweep
;* Description: This subroutine is used to sweep the motion sensor CW then CCW
;*                  as the motion detector is being swept, only the first value obtained
;*                  in each direction of motion is recorded resulting in two values
;*                  one stored in DET_CW, and DET_CCW
;* Uses: TEMP
;* Output: DET_CW, DET_CCW
;*************************************************************************************
Motion_SWEEP:   ;DISP_STRING     nxtline
                cbr STATUS, $08                 ; Clear Status bit 3
                                                ; When Status bit 3 is set then hit recorded
                cbi PORTB, 3
                sbi PORTC, 0
                cbi PORTC, 0

                clr MD_HIT                      ;
                ;mov GENIOI, MD_HIT
                ;rcall HEX2ASCII
                ;DISP_STRING     nxtline

                sbrc STATUS, 4                  ; If Status bit4=1, Jump to CCW_SETUP
                rjmp CCW_SETUP                  ;
CW_SETUP:       ldi GENIOI, MDET_LM             ; else CW_SETUP
                sts MDET_ST, GENIOI             ; Start at Leftmost
                ldi GENIOI, MDET_RM             ;
                sts MDET_END, GENIOI            ; End at RIghtmost
                ;                ;
SWEEP_ST:       rcall TIM2_PWM                  ; Set Timer2 to PWM Mode and PortD OC2 to output
                lds MOT_LOC, MDET_ST            ; Set Servo to Start Position
                out OCR2, MOT_LOC               ;
                ;
                ldi TEMP2, $08                  ; Wait until SERVO SWINGS to Start Position
                rcall Motion_Delay      ;

SWEEP_RST:      rcall CA_HH                     ; Collision Avoidance Human Hit Handler
                ldi GENIOI, $02                 ; Wait for RESET value
                rcall ADC_8bit                  ;
                andi GENIOI, $F0        ;
                cpi GENIOI, $80                 ;
                brne SWEEP_RST                  ;

                sbrc STATUS, 4                  ; If Status bit4=1,CCW
                dec MOT_LOC                     ;       CCW = dec MOT_LOC
                sbrs STATUS, 4                  ; If Status bit4=0,CW
                inc MOT_LOC                     ;        CW = inc MOT_LOC
SWEEP_LP1:      out OCR2, MOT_LOC               ; Incremental Clockwise Servo Swing
                ;
                ldi TEMP2, $03                  ; Delay
                rcall Motion_Delay      ;
                rcall CA_HH                     ; Collision Avoidance Human Hit Handler
                ;
                rcall Motion_Detect     ; Check Motion_Detector
                sbrc STATUS, 3                  ; If Status bit3=1, HIT exit SWEEP
                rjmp SWEEP_EX                   ;
                ;
                sbrc STATUS, 4                  ; If Status bit4=1, CCW
                dec MOT_LOC                     ;       CCW = dec MOT_LOC
                sbrs STATUS, 4                  ; If Status bit4=0, CW
                inc MOT_LOC                     ;        CW = inc MOT_LOC
                ;
                lds TEMP, MDET_END              ; TEMP = MDET_END
                cp TEMP, MOT_LOC                ; Compare TEMP to MOT_LOC
                brne SWEEP_LP1
                ;
                sbrc STATUS, 4                  ; If Status bit 4=1,CCW
                rjmp CCW_CH                     ;       CCW = Jump to CCW_CH
                sbr STATUS, $10                 ; CW = SET STATUS bit 4
SWEEP_EX:       ret
                ;
CCW_CH:                 cbr STATUS, $10                 ;       CCW = CLR STATUS bit 4
                rjmp SWEEP_EX                   ; exit
                ;
CCW_SETUP:      ldi GENIOI, MDET_RM             ; CCW_SETUP
                sts MDET_ST, GENIOI             ; Start at Rightmost
                ldi GENIOI, MDET_LM             ;       ;
                sts MDET_END, GENIOI            ; End at LeftMost
                rjmp SWEEP_ST                   ; Begin Sweep
;*
;*****************************************************************************************
;* Interrupt - External Interrupt 0 Handler
;* Description: When an external interrupt occurs on INT0, a motion is detected by
;*              the sensor. The external Interrupt will check the current direction
;*              and the direction vector to make sure it is the first value for that
;*              direction and if so it records it else it exits
;* Uses: TEMP
;*****************************************************************************************
Motion_Detect:  push TEMP
                ldi GENIOI, $02
                rcall ADC_8bit
                ;push GENIOI
                ;rcall HEX2ASCII
                ;pop GENIOI

                sbrc STATUS, 4                  ; If status bit4=1, CCW
                rjmp MOTDET_CCW                         ;       CCW = Execute CCW Detection

MOTDET_CW:      cpi GENIOI, $92                 ; $90 = Hit @ MDelay=$03
                brmi MOTDET_EX                  ; If GENIOI-$A0 < 0, not a valid value

MD_REC:                 sbr STATUS, $08                 ; set status bit3=1 for hit
```

```
                in TEMP, OCR2                    ; record value into DET_CW

                mov MD_HIT, TEMP                 ;
                sbi PORTB, 3
                sbi PORTC, 0
                cbi PORTC, 0

MOTDET_EX:      pop TEMP
                ret

MOTDET_CCW:     cpi GENIOI, $70                  ;
                brpl MOTDET_EX                   ; If GENIOI-$6B > 0 not a valid value
                rjmp MD_REC                      ; record hit value
```