

Wireless Sensor Network and Laboratories

Final Project Report

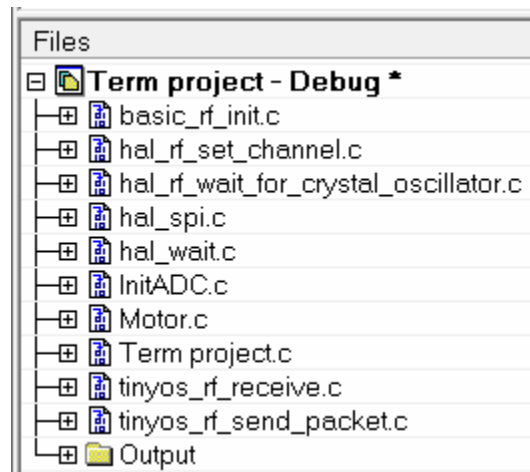
Group 10

GICE Doctor 1 D96942029 王光傳(Thieu Quang Tuan);

ESOE Master 1 R96525069 林佳憲;

ESOE Master 1 R96525071 陳胤辰

1. File Structure



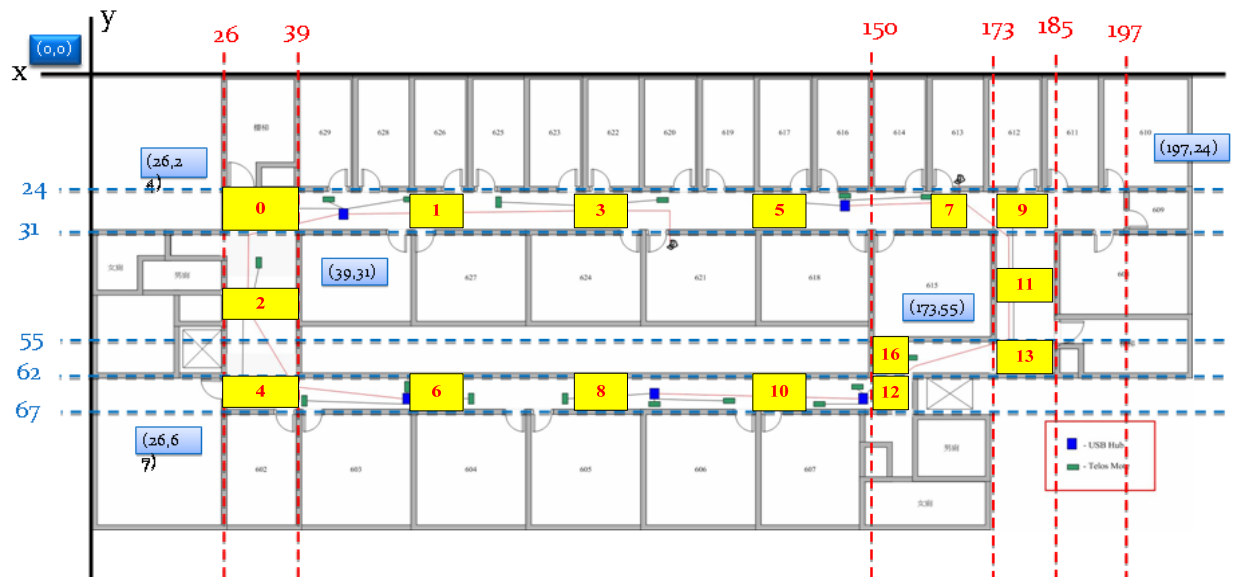
File Name	Description
basic_rf_init.c	Radio
hal_rf_set_channel.c	Radio
hal_rf_wait_for_crystal_oscillator.c	Radio
hal_spi.c	<i>halSpi</i> initial
hal_wait.c	<i>hal</i> wait
Motor.c	Wheel controller.
Term project.c	Main function of our program.
tinyos_rf_receive.c	Radio
tinyos_rf_send_packet.c	Radio

2. System approach -Using location system to go to the destination

2.1. Forwarding Algorithm Introduction

In order to know where we are, we get the location information from the beacon in Localization System. Due to the position of Localization System is not very precise. So we divided the gallery into two kinds of blocks named sensing area and buffer area (see 2.2.) When the car in the buffer area, it just does nothing and uses the avoiding algorithm to keep forwarding. If the car in the sensing area, it will check the sequence of the area number to ensure the way is forwarding to the destination. If the way is correct, it will save new area number and keep going. If it perceived that way is incorrect, it will return 180 degree and keep going (see 2.3.)

2.2. Sensing Map



- ◆ Sensing Area: 0,1,2,3,4,5,6,7,8,9,10,12,12,13,16
- ◆ Buffer Area: Else.

2.3. Check Direction in Sensing Area Pseudo Code

```
If (Car_in_Sensing_Area)
    Check the sequence of the area number;
    If (way is correct)
        forwarding;
    Else
        turn 180 degree;
Else
    Nothing;
```

2.4. *whereWeAre* Pseudo Code

```
If( In_Sensing_Area )
    return Area_Num;
Else
    return -1;
```

◆ *whereWeAre* Code

To flexibility each sensing area size, we define a constant *stateWidth* to set the width of sensing area.

Variable Table:

Type	Variable Name	Description
Input	<i>x, y</i>	Receive coordination.
	<i>stateWidth</i>	Constant, to set size of <i>sensing area</i> .
Output	<i>ret</i>	The number of current area. If car in the <i>buffer area</i> , return -1.

```
int ret = -1;

if(x>=26 && x<=39 && y>=24 && y<=31){
    ret = 0;
}
else if(x>=58 && x<=58+stateWidth && y>=24 && y<=31){
    ret = 1;
}
else if(x>=94 && x<=94+stateWidth && y>=24 && y<=31){
    ret = 3;
}
else if(x>=127 && x<=127+stateWidth && y>=24 && y<=31){
    ret = 5;
}
else if(x>=154 && x<=154+stateWidth && y>=24 && y<=31){
    ret = 7;
}
///////// right-upper corner
else if(x>=170 && x<=183 && y>=24 && y<=31){
```

```

    ret = 9;
}
else if(x>=173 && x<=185 && y>=42 && y<=46){
    ret = 11;
}
///////// right of destination
else if(x>=173 && x<=185 && y>=51 && y<=62){
    ret = 13;
}
//////////////////////////////// bottom-path //////////////////////////////////
else if(x>=26 && x<=39 && y>=46 && y<=46+stateWidth){
    ret = 2;
}
///////// left-bottom corner
else if(x>=26 && x<=39 && y>=62 && y<=67){
    ret = 4;
}
else if(x>=58 && x<=58+stateWidth && y>=62 && y<=67){
    ret = 6;
}
else if(x>=105 && x<=105+stateWidth && y>=62 && y<=67){
    ret = 8;
}
else if(x>=138 && x<=138+stateWidth && y>=62 && y<=67){
    ret = 10;
}
///////// below-destination
else if(x>=154 && x<=162 && y>=58 && y<=67){
    ret = 12;
}
else if(x>=154 && x<=158 && y>=54 && y<=55){
    ret = 16;// destination
}
else{// in the buffer area
    ret = -1;
}

return ret;

```

2.4. Go to Destination Pseudo Code

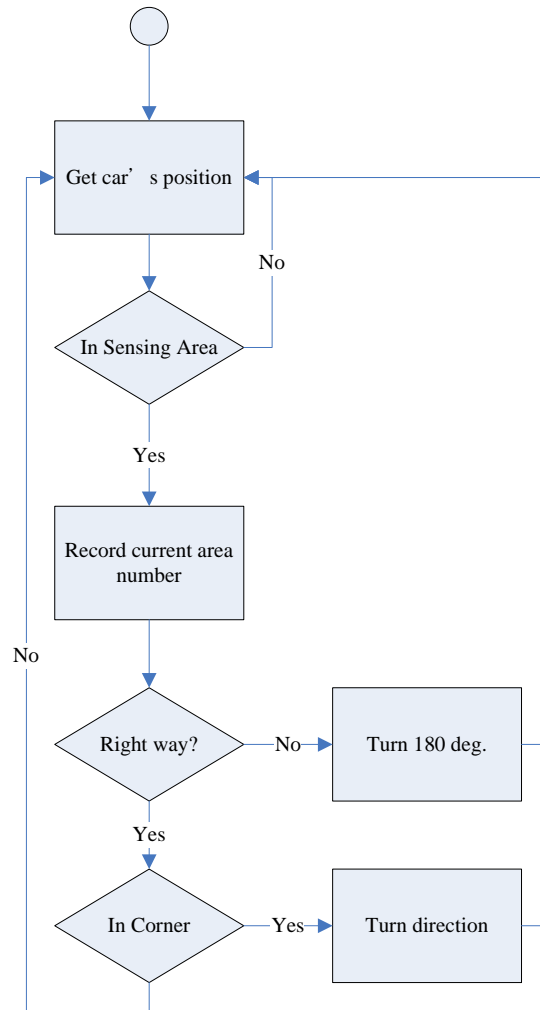
```
x = Receive_pkt_position_x;
y = Receive_pkt_position_y;

nowState = Initial_Position_number;           // set initial state
Forwarding();                                // with avoiding obstacle algorithm

here = whereWeAre(x,y) ;

if( here != -1 ){                             // in the sensing area
    if(here != nowState){
        if ( here == nextState(nowState)){    // correct direction
            nowState = here;
            if( here == Destination_num ){
                Stop_car();
            }
        }
        else if( here == previousState (nowState)){ // wrong direction
            nowState = here;
            Turn_180_degree();
        }
    }
}
```

2.5. Go to Destination Flow Chart



2.6. The Key Method of Forwarding to the Destination

We Split the gallery into several blocks, giving each block a sequence number from 0 to 13 and the destination sequence number is 16(see the yellow block in fig 2.2). And the blocks that don't have number is buffer area.

In the initial step, we install the start and destination block into Taroko.

In the running step, the car will transform the location information into block number, and use these block number to check whether the direction that the car goes is correct or not. For example, if the car in the block 9(sensing area) now, in a moment, it receive the location to know it is in the block 7 or less than 9, then the car will perceive it goes the wrong direction and turn 180 degree(line 27~30) .Not until the car in the block 16(destination),it will keep forwarding and avoiding obstacle(line 10~13、22~24).

If the car in the buffer area, then it will keeping forwarding and skip the location packet (line 2、13).

In order to turn the correct direction to forward to the destination, so we define some special area in the corner(ex: block 4、9、12、13). When the car in these special area, it will turn some degree to go to the destination more smoothly. We use a flag to decide whether the car need to turn some degree or not (line 18~20). For example, when the car in block 9 and block 13, it will turn right 45 degree and 90 degree.

```
1.         here = whereWeAre(Xnow, Ynow);
2.         if(here != -1){           // here == -1 is means car in the buffer space
3.             if(here != nowState){ // when car run to another state
4.                 if(nowState == 0){ // the car in the state 0
5.                     if( here == 1 || here == 2){ // so the next state can be 1 or 2
6.                         nowState = here;
7.                     }
8.                 }
9.             else if(nowState == 13){ // the car in the state 13
10.                if( here == 16 || here == 12){ // so the next state can be 16
11.                    nowState = here;
12.                    Stop();
13.                    holdTime = 1000;
14.                }
15.            }
16.            else if(here == nextState(nowState)){ // right direction
17.                nowState = here;                // renew nowState
18.                if((nowState == 4) || (nowState == 9) || (nowState == 12) ||
19.                (nowState == 13)){
20.                    fNeedTurn = TRUE;           //flag to control turn direction
21.                }
22.                else if(nowState == 16){//
23.                    Stop();
24.                    holdTime = 1000;
25.                }
26.            }
27.            else if(here == prevState(nowState)){ // wrong direction
28.                // turn 180
29.                fWrongWay = TRUE;
30.                nowState = here;
31.            }
32.        }
33.    } //here== -1, means car in buffer space, keep going
```

3. Setup the Infrared Sensor

3.1. Infrared Sensor Threshold Setting



Fig.3.1

We find out the relation between the value of the infrared sensor and distance from car to obstacle by ruler(see Fig.3.1).

3.2. Infrared Sensor Placement

We use several methods to put the infrared sensor. We try to use Styrofoam 、Pipette and Two-side Sticky.

3.2.1. Use Styrofoam(保麗龍)



Introduction:

First, we want to use something to represent the car's eyes. We use two Styrofoam to as the base of infrared sensor, and setup the sensor on it.

Drawback:

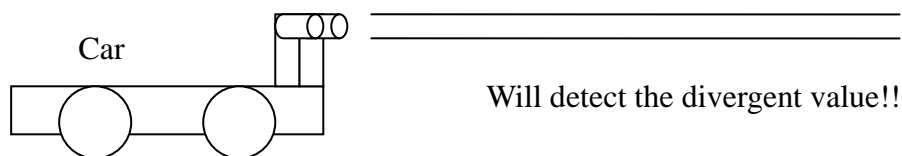
1. using Styrofoam as Base is Unstable

The two "eyes"(or sensor) are unstable. Because the Styrofoam is not strong enough, the sensor will vibrate when the car is moving. It will cause the sensor to get the wrong value.

2. Value Divergence

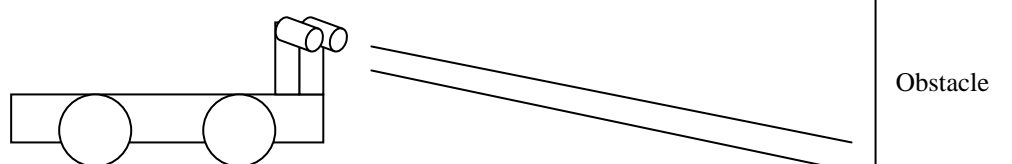
Another, we find out some problem of the infrared sensor. When the sensor saw more than 70cm, the sensing value seems to be divergent.

Let the infrared sensor and ground be parallel



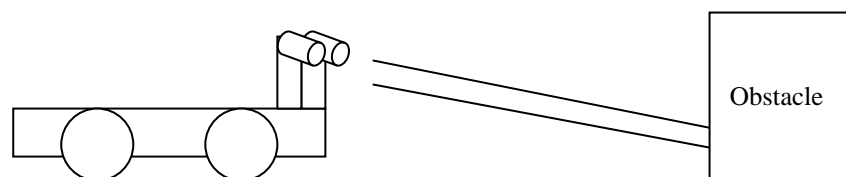
So we want to let the sensor to see the ground, as following picture.

Before detecting the obstacle



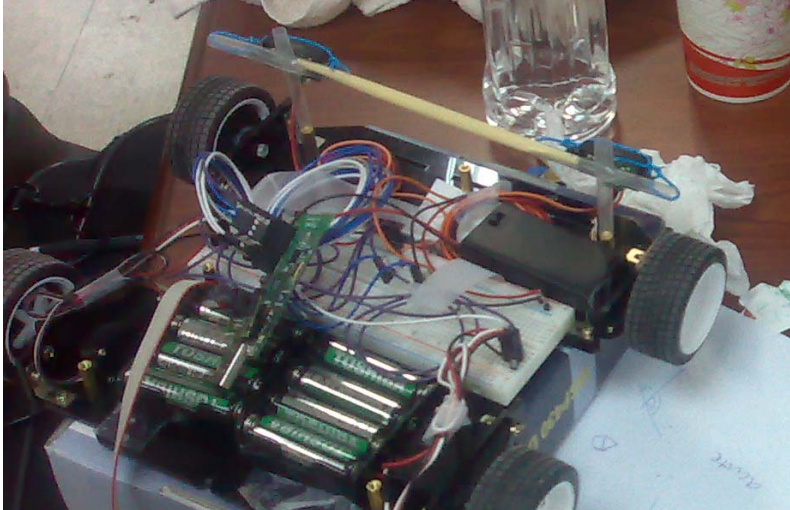
Before detecting the obstacle, the value sensing by sensors will keep in some fixed range.

After detecting the obstacle



When detecting the obstacle, the value sensing by sensors will effect by obstacle.
So we improve the angle of the deployment of the infrared sensor.

3.2.2. Using Pipette(吸管)



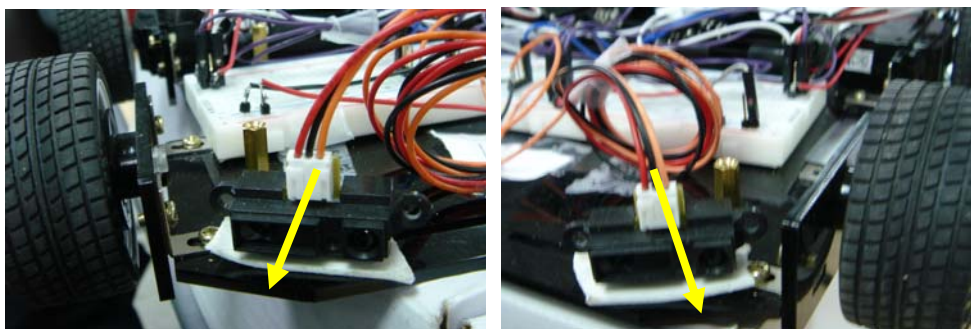
Introduction:

We use pipette and chopstick to be a backbone to setup the infrared sensors. This method will let the sensor more stable and also keep the sensor to see the ground easily.

Drawback:

1. Still unstable.
2. The position of the Infrared sensor is too high to see the obstacle near the ground.

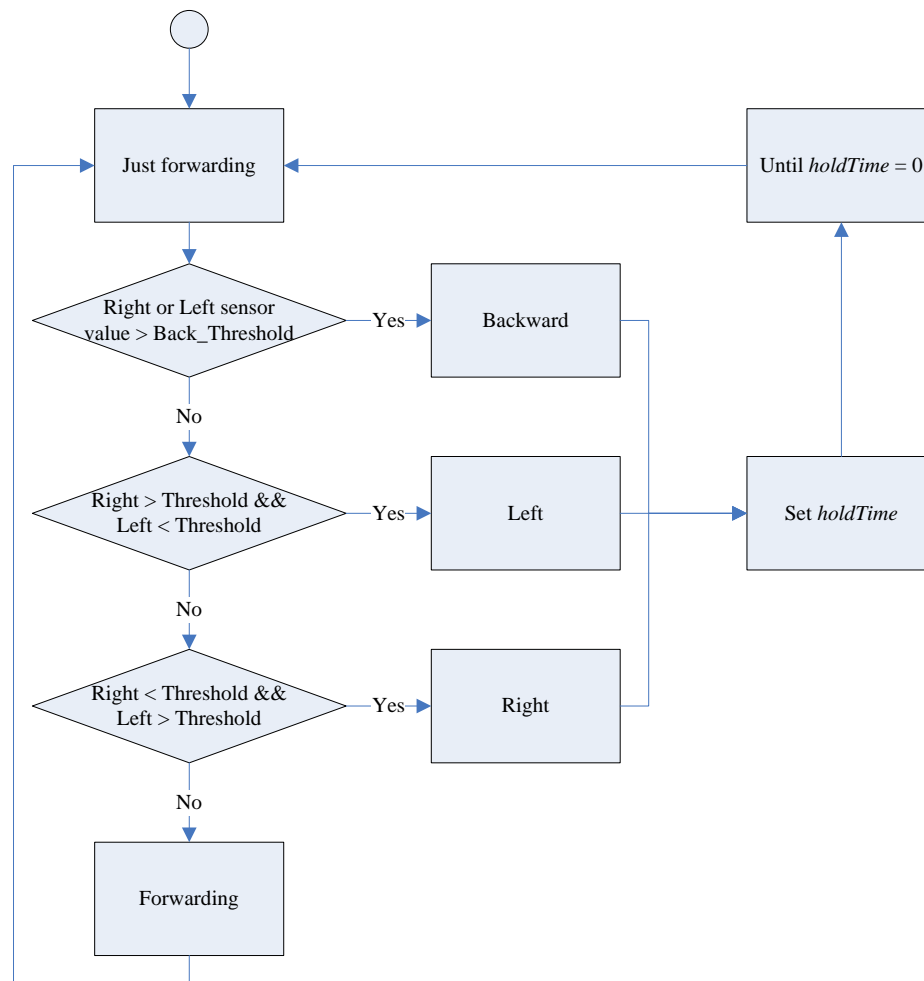
3.2.3. Using Two-side Sticky(黏雙面膠)



Finally, we reference other group's method. We use two-side sticky to stick the infrared sensor in the front of the car, and let the sensing scope wider. After finishing the setup, we find this method has higher performance to avoid the obstacle. So we adopt this method.

4. Avoiding Obstacle Algorithm

4.1. Avoiding Obstacle Algorithm Flow Chart

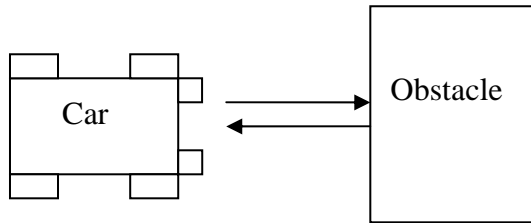


4.2. Deadlock Detection

In the indoor, the topography is very complicated. Sometimes, when the car meet some special situation (ex: wall ,corner), it's sensing value will fall into some deadlock decision. In this way, the car will not escape this forever loop. In order to avoid this situation, we use two strategies to resolve it.

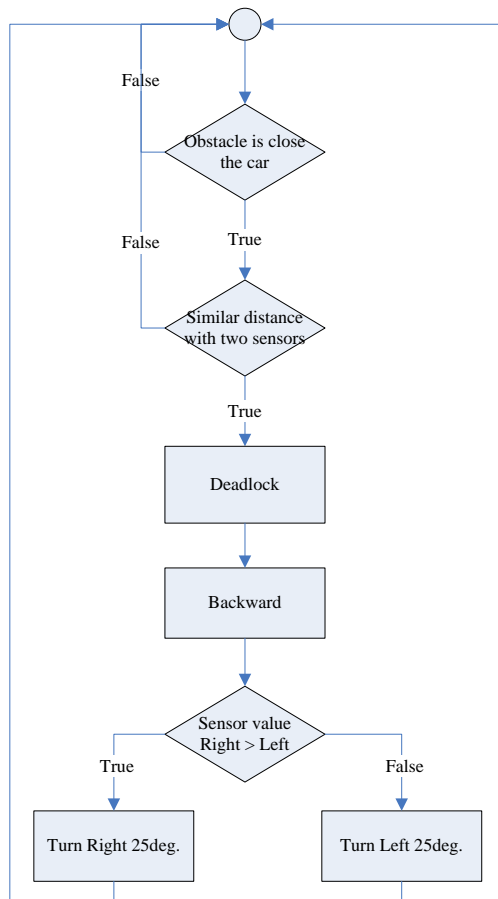
4.2.1. Repeat Forwarding and Backward(when meeting a wall)

Situation: Car just forwarding and backward, when meeting a wall.

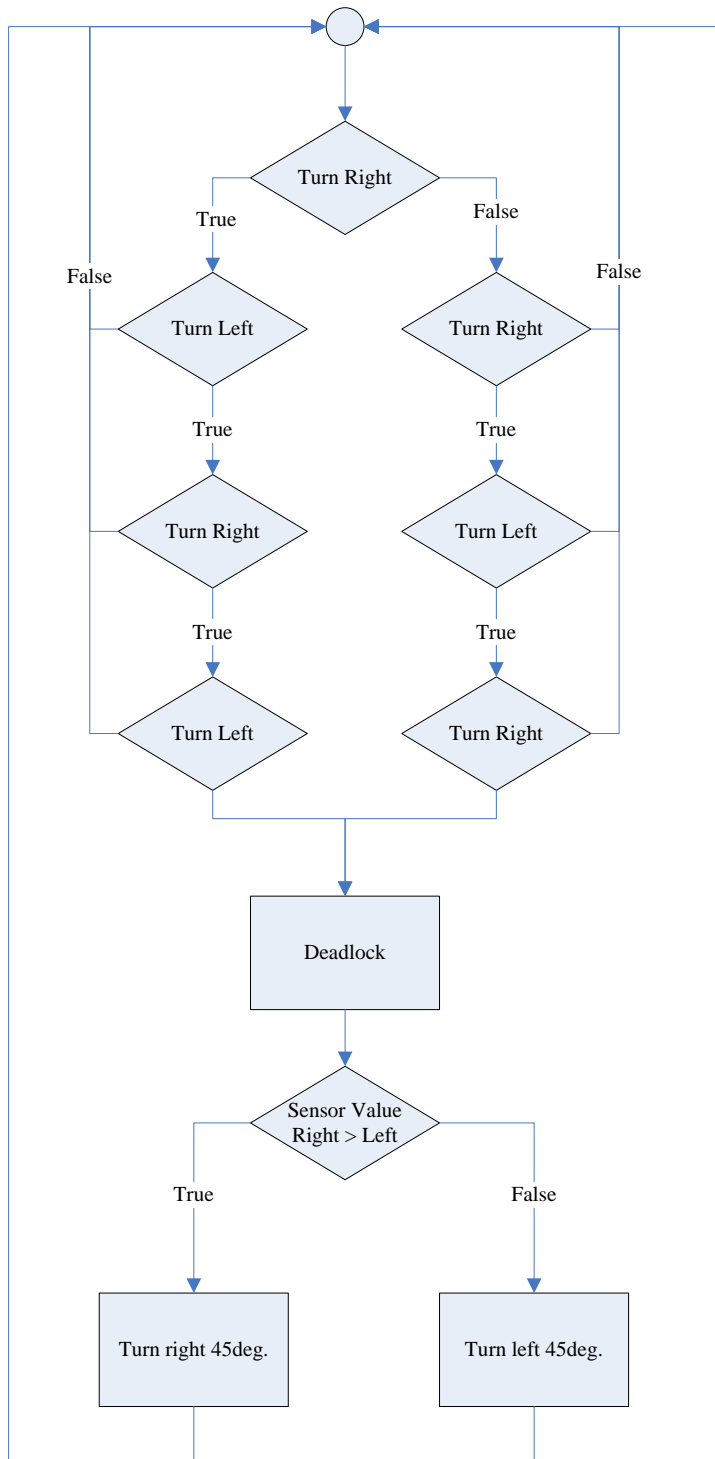


Resolution: When two infrared sensors detect the approximate value and these two values also over the threshold, the car will backward first, then using distance detected latest to decide to turn right or turn left a small angle. For example, when the car meets a wall, the distance from right sensor to the car is a little more than left, these two values both over the threshold, then the car will backward first and turn right a small angle.

Flow Chart



4.2.2. Repeat Turn Left and Turn Right Flow Chart(when go into the corner)



When the car go into the corner, it will continue turn right and turn left. So we use a counter to calculate the frequency of the car turning times. For example, when the car goes into a corner, and it tries to turn left to escape this corner. After turning left, it will detect another wall and try to turn right. The same as we mentioned before, it will turn right and enter into a deadlock. When the turning times are counted to 4, the car will choose turn right to escape the corner.

5. Controlling Robot Car

5.1. Wheel Control

5.1.1. Timer Control

We use *Timer B* to control the motor.

Timer B1: control left wheel.

Timer B2: control right wheel.

5.1.2. Forwarding

When we want car forwarding, we should let two motors rotate in the different way.

Therefore, we let right side wheel turns clockwise, and left side wheel turns counter-clockwise.

```
void Forward(void)
{
    P2DIR |= BIT3+BIT6;           // Set P2.3,6=1 --> output direction
    P2SEL &= ~(BIT3+BIT6);
    // Set P2.3,6=0 --> function as I/O to control motor

    TBCTL |= TBCLR + MC_0;
    TBCTL &= ~TBIE;

    TBCCR0 = 655;
    TBCCTL0 = CCIE;              // CCR0 toggle, interrupt enabled

    TBCCR1 = 655+CounterClockwise;
    //to control left wheel ,>50 =>counter-clockwise(forward)
    TBCCTL1 = CCIE;             // BIT4=CCIE, CCR1 toggle, interrupt enabled

    TBCCR2 = 655+Clockwise;
    //to control right wheel ,<50 =>clockwise(forward)
    TBCCTL2 = CCIE;            // BIT4=CCIE, CCR1 toggle, interrupt enabled

    //CCTL2 = OUTMOD_4 + CCIE;    // CCR2 toggle, interrupt enabled
    TBCTL = TBSSEL_1 + MC_2 + TBIE; // SMCLK, Contmode, int enabled
}
```

5.1.3. Turn Left and Turn Right

If two wheels rotated in the same way, then the car will turn right or turn left.

Turn Left

```
...
    TBCCR1 = 322+Clockwise; // left wheel
...
    TBCCR2 = 322+ Clockwise; // right wheel
...
```

Turn Right

```
...
    TBCCR1 = 322+ CounterClockwise; // left wheel
...
    TBCCR2 = 322+ CounterClockwise; // right wheel
...
```

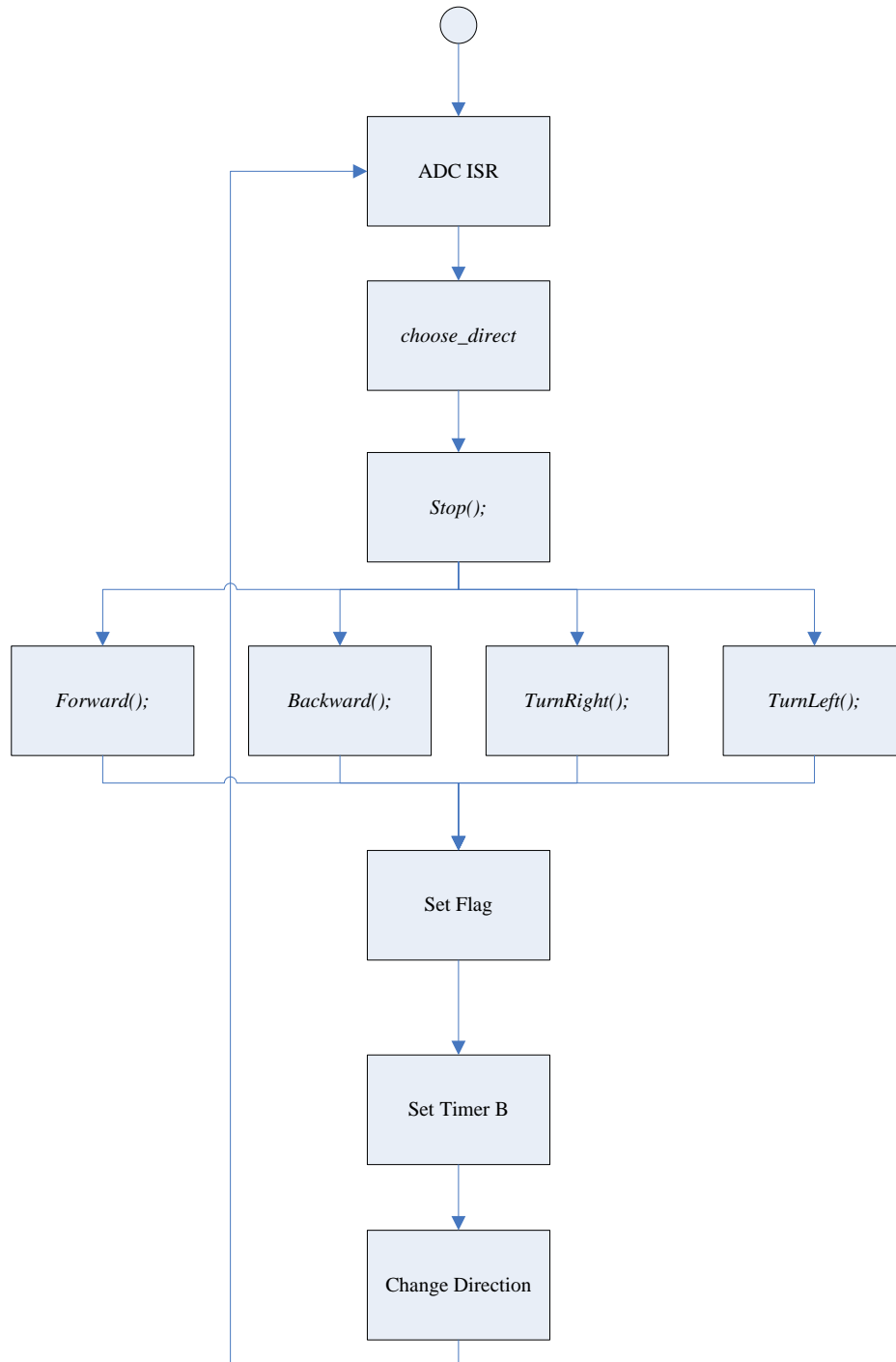
5.1.4. Backward

Opposite way to forwarding, the car will move backward.

```
...
    TBCCR1 = 322+Clockwise; // left wheel
...
    TBCCR2 = 322+CounterClockwise; // right wheel
...
```

5.2. Turning Control Flow Chart

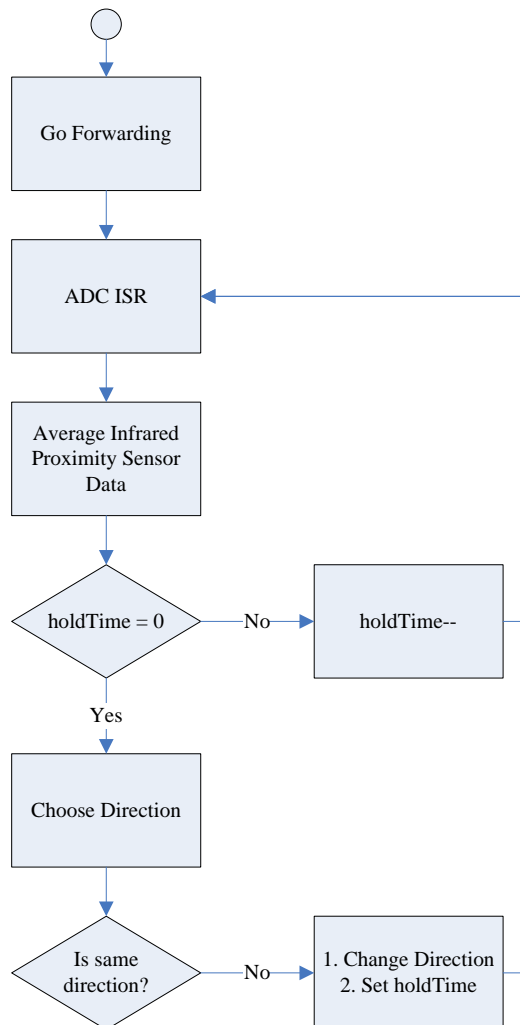
We set some flag to know what the car status now. When the program decides the car needs to change another direction, it will look flags to determine it need to change or not.



5.3. Keep turning

We use a variable *holdTime* to keep the turning state and extend the turning time. If the *holdTime* is equal to 0 and the car need to turn another direction, then the car can change the turning direction.

Change Direction Flow Chart



5.4. Turning Angle

Using try and error to know the relation between *holdTime* and angle. And get the formula.

```
int setTurnDeg(int m_deg){ // degree

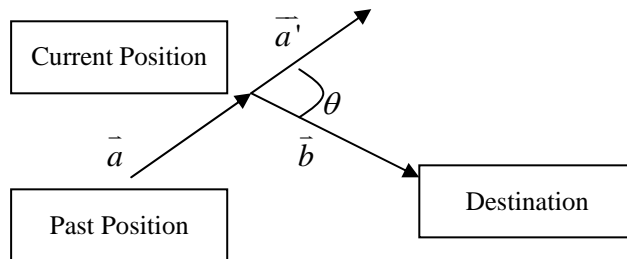
    return (int)((m_deg/2)-7);

// input 90(deg), it will return holdTime = 47
}
```

6. Fault Approach - Vector Algorithm

6.1. Idea

We used three-points to create two-vectors to calculate which degree should be turned.



Vector \vec{a} can translation to \vec{a}' . We assume $\theta = \cos^{-1} \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$

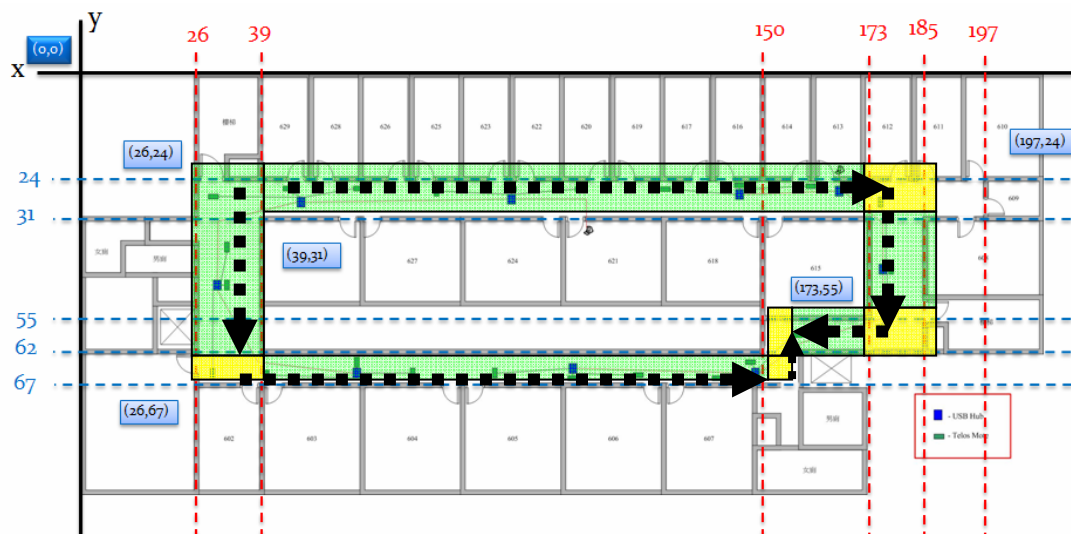
6.2. Calculate Two Vector Angle

- Past Position: (a_0, a_1)
- Current Position: (b_0, b_1)
- Destination Position: $(destX, destY)$

```
int CalAngle(int a0, int a1, int b0, int b1){
    // a0,a1 previous localtion
    // b0,b1 now localtion
    double deg;
    double distA,distB; //distance
    int vctA0,vctA1,vctB0,vctB1; // vector A=(vctA0,vctA1);B=(vctB0,vctB1)
    vctA0 = destX-b0;
    vctA1 = destY-b1;
    vctB0 = b0-a0;
    vctB1 = b1-a1;

    distA = sqrt(pow(destX-b0,2)+pow(destY-b1,2));
    distB = sqrt(pow(b0-a0,2)+pow(b1-a1,2));
    if((distA != 0) || (distB !=0))
    {
        deg = acos( (vctA0*vctB0+vctA1*vctB1)/(distA*distB));
        deg=deg*180/___PI;
    }
    else
        deg=0;
    return (int)deg;
}
```

6.3. Set Destination



When car is in green region, we set the destination in the arrow direction region.

```

Void setDest(BYTE x, BYTE y){// setDest
    BOOL ret;
    if (x>39 && x<197 && y>24 && y<31){ // upper band
        destX=180;
        destY=28;
    }
    else if (x>26 && x<39 && y>31 && y<62){ // left band
        destX=31;
        destY=65;
    }
    else if (x>173 && x<185 && y>31 && y<55){ // right band
        destX=180;
        destY=58;
    }
    else if (x>39 && x<150&& y>62 && y<67){ // under band
        destX=155;
        destY=65;
    }
}
    
```

6.4. Check car is or not in the corner

If car is in the corner, we set the next destination coordination (destX/destY).

```
BOOL isInConner(BYTE x, BYTE y){// setDest
  BOOL ret;
  if (x>26 && x<39 && y>24 && y<31){ // initial conner
    destX=32;
    destY=65;
    ret= TRUE;
  }
  else if (x>26 && x<39 && y>62 && y<67){ // left-under conner
    destX=155;
    destY=65;
    ret= TRUE;
  }
  else if (x>173 && x<185 && y>24 && y<31){ // right-upper conner
    destX=180;
    destY=58;
    ret= TRUE;
  }
  else if (x>150 && x<173&& y>62 && y<67){ // under-destination conner
    destX=destXfinal;
    destY=destYfinal;
    ret= TRUE;
  }
  else if (x>173 && x<185 && y>55 && y<62){ // right-destination conner
    destX=destXfinal;
    destY=destYfinal;
    ret= TRUE;
  }
  else if (x>185 && x<197 && y>24 && y<31){ // right of right-upper conner
    destX=180;
    destY=28;
    ret= TRUE;
  }
  else{ // just go
    ret= FALSE;
  }
  return ret;
}
```

7. Conclusion

We use a lot of time to implement this laboratory. We spend almost time to test infrared sensor and improve all problem. In the last day we take over 6 hours to check Bolly building's position location, to make sure our car can determine correct direction by the localization system. Finally, we succeed to go to the destination. But we miss the correct packet to let the car stop. In spite of this fault, we still learning a lot in this class and also gained a memorable friendship.



From left to right: Jason, SY, Michael, Tuan