```
#use "cmucamlib.ic"
#define CLICKS_0 24.5   //left  12
#define CLICKS_1 24.5   //right 13
#define ENCODER_0 2 //left
#define ENCODER_1 3 //right
#define IR_SENSOR1 2
#define IR_SENSOR2 4
#define L_MOTOR 0
#define R_MOTOR 2
#define MAX 222
#define PI 3.1415
#define THRESHOLD 0.1
#define turn_180 1.8
#define turn_90 0.57

float x[21];  // x cordinates in the WM
float y[21];  // y coordinate sin the WM
float dist_r_t[16];  //distance between robo and the cubes
float dist_r_d[4];   //distance between robo and the destinations
float dist_r_t_x[16];//x-distance between robo and the cubes
float dist_r_t_y[16];//y-distance between robo and the cubes
float dist_r_d_x[4]; //x-distance between robo and the destinations
float dist_r_d_y[4]; //y-distance between robo and the destinations
float min_dist_x;//x-distance between robo and the nearest cube or
destination
float min_dist_y;//y-distance between robo and the nearest cube or
destination
float r_x;// current x-coordinate of the robo
float r_y;//current y-coordinate of the robo


int blue;
int orange;
int orange_size;
int target=0;
int n,s,e,w; // directions
int current_t,current_d; //current array index of the cube(target(_t))
or dest(_d)where the robo is nearest
int distance_t_pid=0,color_pid=0,tape_pid=0,cam_pid=0;
int s1=0; //ir left sensor bool value set to 1 when it reaches black
tape
int s2=0;//ir right sensor bool value set to 1 when it reaches black
tape
int left=0;
int right=0;

void direction(int a,int b,int c,int d)  // gives the direction in
which the robot is heading
{
    n=a;
    s=b;
    e=c;
    w=d;
}

void move_t()   // moves the robo to the nearest cube or to the nearest
destination
```

```
{
    int left,right;
    if(min_dist_x<0.0 && min_dist_y<0.0)
      {
        printf("minors 1\n");
        sleep(1.0);
        TURN(1);
        while(read_encoder(3)<=35);
        ao();
        printf("minors 2\n");
        sleep(1.0);
        FORWARD(100);
        direction(1,0,0,0);
        encoder_dist(min_dist_y,&r_y);
        ao();
        TURN(1);
        while(read_encoder(3)<=35);
        ao();
        FORWARD(100);
        direction(0,0,1,0);
        encoder_dist(min_dist_x,&r_x);
        ao();
    }
    if(min_dist_x<0.0 && min_dist_y>0.0)
      {
        TURN(-1);
        while(read_encoder(3)<=34);
        ao();
        FORWARD(100);
        direction(0,1,0,0);
        encoder_dist(min_dist_y,&r_y);
        ao();
        TURN(-1);
        while(read_encoder(3)<=34);
        FORWARD(100);
        direction(0,0,1,0);
        encoder_dist(min_dist_x,&r_x);
        ao();
    }

    if(min_dist_x>0.0 && min_dist_y>0.0)
      {
        FORWARD(100);
        direction(0,0,0,1);
        encoder_dist(min_dist_x,&r_x);
        ao();
        msleep(1000L);
        TURN(-1);
        while(read_encoder(3)<=34);
        ao();
        msleep(1000L);
        FORWARD(100);
        direction(0,1,0,0);
        encoder_dist(min_dist_y,&r_y);
        ao();
    }
    if(min_dist_x>0.0 && min_dist_y<0.0)
```

```
      {
        FORWARD(100);
        direction(0,0,0,1);
        encoder_dist(min_dist_x,&r_x);
        ao();
        TURN(1);
        while(read_encoder(3)<=34);
        ao();
        FORWARD(100);
        direction(1,0,0,0);
        encoder_dist(min_dist_y,&r_y);
        ao();
    }
    // ao();

}

void distance_t()   //calculates the nearest target object from the
actual position of the robot
{
    int i=0;
    float min_dist=100.0;
    int temp=0;
    printf("manohar\n");
    while(!stop_button()) {
        // target=0;
        min_dist=100.0;
        for(i=0;i<16;i++)
          {
            if(x[i] > 0.0) {
                dist_r_t_x[i]=x[i]-r_x;
                dist_r_t_y[i]=y[i]-r_y;


dist_r_t[i]=sqrt((dist_r_t_x[i]*dist_r_t_x[i])+(dist_r_t_y[i]*dist_r_t_
y[i]));
                //if(i==0)
                //  min_dist=dist_r_t[0];
                if(min_dist>dist_r_t[i])
                  {
                    min_dist=dist_r_t[i];
                    min_dist_x=dist_r_t_x[i];
                    min_dist_y=dist_r_t_y[i];
                    current_t =i;
                }
            }
            else{
                dist_r_t_x[i] = 100.0;
                dist_r_t_y[i] = 100.0;
                dist_r_t[i] = 1000.0;
            }

        }
        move_t();

    }
}
```

```
void distance_d()   //calculates the nearest destination square from
the actual position of the robot(when if has a cube with it)
{
    int i=0;
    float min_dist=100.0;
    for(i=0;i<4;i++)
      {
        if(x[i+16] >= 0.0) {
            dist_r_d_x[i]=x[i+16]-r_x;
            dist_r_d_y[i]=y[i+16]-r_y;


dist_r_d[i]=sqrt((dist_r_d_x[i]*dist_r_d_x[i])+(dist_r_d_y[i]*dist_r_d_
y[i]));
            //if(i==0)
            //  min_dist=dist_r_d[0];
            if(min_dist>dist_r_d[i])
              {
                min_dist=dist_r_d[i];
                min_dist_x=dist_r_d_x[i];
                min_dist_y=dist_r_d_y[i];
                current_d=i;
            }

        }
        else{
            dist_r_d_x[i] = 100.0;
            dist_r_d_y[i] = 100.0;
            dist_r_d[i]=1000.0;
        }


    }
    move_t();    //after calculating the distance the robot moves to
the destnation square
}

void m_initialize()       // my initialization.. set every thing to -1-
1
{
    int i=0;
    for(i=0;i<21;i++)
      {
        x[i]=-1.0;
        y[i]=-1.0;
    }

    for(i=0;i<16;i++)
      {
        dist_r_t[i]=0.0;
        dist_r_t_x[i]=0.0;
        dist_r_t_y[i]=0.0;
    }

    for(i=0;i<4;i++)
      {
        dist_r_d[i]=0.0;
        dist_r_d_x[i]=0.0;
        dist_r_d_y[i]=0.0;
    }

                               //Dr.hougen assigns the coordinates in
h_initialize()

    h_initialize();             //in this initialize r_x,r_y(current
coordinates of the robo)
    m1_initialize();
}

void encoder_dist(float x,float *y)  //keeps track of the distance the
robo has travelled
{                                    // in X or Y direction
    float dist;
    float reading;
    while(1)
      {
        dist = ((float)read_encoder(2)+(float)read_encoder(3))/49.0;
        if(x<0.0)
          {
            if(dist>=-x)
              break;
        }
        else
          {
            if(dist>=x)
              break;
        }
    }

    //dist = reading*PI*DIAM;
    if(x>0.0)
      *y = *y + dist;
    else
      *y = *y - dist;
    printf("distance = %f\n", dist);
    sleep(1.0);
}

void FORWARD(int x)// makes the robot go in straight line at x velocity
{
    int y;
    y=(int)(((float)x)*0.85); // Was done in calibration
    reset_encoder(2);
    reset_encoder(3);
    enable_encoder(2);
    enable_encoder(3);
    motor(L_MOTOR,y);
    motor(R_MOTOR,x);
}

void TURN(int dir)// makes the robot turn 90 or 180 according with dir.
{
    reset_encoder(2);
    reset_encoder(3);
    enable_encoder(2);
    enable_encoder(3);
    motor(L_MOTOR,96*dir);
    motor(R_MOTOR,-100*dir);
}

void color()               //this function checks if we have a target
near
{                          // and alos for the mobile robot
    int i=0;
    float a,b;
    while(!stop_button()) {
        if( orange>60)// && orange_size > 20)    //if the robo doesn't
have any cube with if and found a new cube
          {                          //take the cube to nearest
dest(call distance_d())
            printf("Target found!\n");
            kill_process(distance_t_pid);
            ao();
            target=1;
            if(n>0)
              {
                TURN(-1);
                while(read_encoder(3)<=32);
                direction(0,0,0,1);
                ao();
            }
            if(s>0)
              {
                TURN(1);
                while(read_encoder(3)<=32);
                direction(0,0,0,1);
                ao();
            }
            if(e>0)
              {
                TURN(1);
                while(read_encoder(3)<=66);
                direction(0,0,0,1);
                ao();
            }
            a = x[current_t]-r_x;
            b = y[current_t]-r_y;
            if((a<.6&&a>-0.6)&&(b<.6&&b>-.6))// updare the WM .if we
had the coordinates of the found cube
              {                              //assign(-1,-1) to say
that we do not have any cube there.
                x[current_t]=-1.0;
                y[current_t]=-1.0;
            }
            distance_d();

        }
        if(target>0 && orange>60)  //if the robot has a cube and found
a new cube on its way to dest
```

```
          {                        //just update the WM saying we have
a cube at that coordinates
            int i=0;
            for(i = 0;i<16;i++)
              {
                if(x[i]==-1.0)
                  {
                    x[i] = r_x;
                    y[i] = r_y;
                    break;
                }
            }
        }
        if(blue>60)
          {
            kill_process(distance_t_pid);//track the mobile thing
            FORWARD(100);
            if(n>0)
              {
                encoder_dist(-1.0,&r_y);
            }
            if(s>0)
              {
                encoder_dist(1.0,&r_y);
            }
            if(e>0)
              {
                encoder_dist(-1.0,&r_x);
            }
            if(w>0)
              {
                encoder_dist(1.0,&r_x);
            }
            //sleep(1.5);
            ao();
            //if(n>0)
            //{
            //TURN(100);
            //sleep(turn_90);
            //}
            //if(s>0)
            //{
            //TURN(-100);
            //sleep(turn_90);
            // }
            // if(e>0)
            // {
            // TURN(100);
            // sleep(turn_180);
            //}
            distance_t_pid = start_process(distance_t());

        }
    }
}

void tape()// this function checks for a black tape
```

```
{
    int i = 0;
    float a,b;
    s1=0;
    s2=0;
    while(!stop_button()) {

        if(analog(IR_SENSOR1)>MAX)
          {
            if(color_pid!=0)
              {
                kill_process(color_pid);
                color_pid=0;
            }
            ao();
            s1=1;
            calibrate();
        }
        else
          if(analog(IR_SENSOR2)>MAX)
            {
              if(color_pid!=0)
                {
                  kill_process(color_pid);
                  color_pid=0;
              }
              ao();
              s2=1;
              calibrate();
          }

          if(s1==1 && s2==1)
          {

            if(target>0)            //dump the target in the
destination
              {
                target=0;
                FORWARD(100);
                sleep(0.3);
                ao();
                FORWARD(-100);
                sleep(0.3);
                beep();
                beep();
            }
            a = x[current_d]-r_x;
            b = y[current_d]-r_y;
            if((a<.6&&a>-0.6)&&(b<.6&&b>-.6)) // if we reach the
destination which is already in the WM
              {
            }
            else//update the coordinates of destination(since it is not
in the WM)
              {
                for(i = 0;i<4;i++)
                  {
                    if(x[i+16]==-1.0)
                      {
                        x[i+16] = r_x;
                        y[i+16] = r_y;
                        break;
                    }
                }
            }
            if(n>0)                 //the below 3 if statements makes
sure that we
              {                     //the robot always faces west
                TURN(-1);
                while(read_encoder(3)<=32);
                direction(0,0,0,1);
                ao();
            }
            if(s>0)
              {
                TURN(1);
                while(read_encoder(3)<=32);
                direction(0,0,0,1);
                ao();
            }
            if(e>0)
              {
                TURN(1);
                while(read_encoder(3)<=66);
                direction(0,0,0,1);
                ao();
            }
            printf("chandu\n");
            sleep(1.0);
        }// end if s1 =1 and s2 = 1


        if(distance_t_pid==0)
          distance_t_pid = start_process(distance_t());
        if(color_pid==0)
          color_pid = start_process(color());
        s1=0;
        s2=0;
    }
    }

void calibrate() // makes sure the robot is perpendicular to the tape
{                // only works when the robot moves in such a way that
    if(s1>0&&s2<1)//both the sensors detect the tape(within 1ft of tape
lenght)
      {

        sleep(3.0);
        while(analog(IR_SENSOR2)<MAX) //when the left sensor sees the
tape first
          {
            motor(R_MOTOR,30);
            msleep(10L);
            ao();
        }
        s2=1;
        ao();
        while(analog(IR_SENSOR1)<MAX)
          {
            motor(L_MOTOR,-20);
            msleep(10L);
            ao();
        }
    }
    if(s1<1&&s2>0)//when the right sensor sees the tape first
      {

        sleep(2.0);
        while(analog(IR_SENSOR1)<MAX)
          {
            motor(L_MOTOR,30);
            msleep(10L);
            ao();
        }
        s1=1;
        ao();
        while(analog(IR_SENSOR2)<MAX)
          {
            motor(R_MOTOR,-20);
            msleep(10L);
            ao();
        }
    }

}


void cam() //  always keeps track of the coloured objects ahead(blue and orange)
{
    // init_camera();
    // clamp_camera_yuv();
    while(!stop_button()) {
        //trackRaw(230,250,60,160,10,20);
        // if(target==0)
        // {
        track_orange();
        orange=track_confidence;
        orange_size = track_area;
        //}
        if(orange_size > 0 && orange > 60) {
            //target=1;
            printf("saw orange\n");
            sleep(0.5);
        }
        else
          {
            // sleep(1.0);
            // printf("nothn\n");
            // orange=0;
            blue=0;
        }

        //trackRaw(140,240,190,230, 60,110);
        track_blue();
        blue=track_confidence;
        if(blue > 60) {
            printf("saw blue\n");
            sleep(0.5);
        }
    }
}

void h_initialize() //Initialize the world model
{
    //   x[20] = 0.0;
    //   y[20] = 0.0;

    x[0] = 7.0;
    y[0] = 3.0;
    x[1] = 2.0;
    y[1] = 3.5;
    x[2] = 6.0;
    y[2] = 4.5;
    x[3] = 6.0;
    y[3] = 6.0;
    x[4] = 1.0;
    y[4] = 7.0;
    x[5] = 1.0;
    y[5] = 9.5;
    x[6] = 6.5;
    y[6] = 10.0;
    x[7] = 7.5;
    y[7] = 10.0;

    x[20] = 4.0;
    y[20] = 5.5;
    x[16]= 1.0;
    y[16]=1.0;
    x[17]=6.0;
    y[17]=3.0;
    x[18] = 2.0;
    y[18] = 11.0;

    r_x = x[20];
    r_y = y[20];

}

void m1_initialize()
{
    int i=0;
    //for(i=0;i<16;i++);
    //{
      //  if(x[i]<1.0||y[i]<1.0||x[i]>11.0||y[i]>7.0)
        //  {
          //  x[i]=-1.0;
            //y[i]=-1.0;
```

```
       // }
//    }
}

void main()
{
    init_camera();
    clamp_camera_yuv();
    printf("done!!!! \n");
    // This makes sure that the robot doesn'n start untill start button
is //pressed
    while(!start_button());
    m_initialize();
    distance_t_pid = start_process(distance_t());
    color_pid = start_process(color());
    tape_pid = start_process(tape());
    cam_pid = start_process(cam());

    while (!stop_button()) ;
    kill_process(distance_t_pid);
    kill_process(color_pid);
    kill_process(tape_pid);
    kill_process(cam_pid);

}
```