UNIVERSITAT DE BARCELONA

Objectius projecte

- Ha d'anar a buscar la paret més propera.
- Un cop trobada, l'han d'anar resseguint sense en cap cas col·lidir amb ella. Ha de mantenir sempre una distància de seguretat de com a mínim 2 mm i en cap cas allunyar-se d'aquesta més de 10 mm.
- Haurà de superar les diferents corbes possibles que es pot trobar en les cantonades. Opcionalment, també ha de ser capaç de sortir d'un cul de sac tenint en compte el interval de distàncies màxim i mínim especificat anteriorment.

Canvis conceptuals

- Disposem de 3 threads:
  - El codi del robot (main loop).
  - El d'emulació del joystick i botons (i.e. lectura del teclat)
  - El d'emulació dels mòduls dynamixels, que ara també actua com a simulador.

- Tasques del simulador:
  - Llegir la velocitat actual dels motors en mode endless turn.
  - Actualitzar l'orientació i posició del robot
  - Verificar no em sortit de l'habitació
  - Verificar no hi ha hagut col·lisió
  - Actualitzar els valors del sensor
  - Finalitzar la simulació després d'un determinat nombre de passos

Idealitzacions del simulador

- El robot és puntual
- El valor del sensor d'infraroig de distància no te error i dona directament el valor de distància (el infraroig dona el valor de llum reflectida i per tant va al inversa).
- L'habitació per on és mou el robot fa 4095 x 4095 mm i te una resolució de 1 mm.
- La velocitat màxima és tal que en un pas del simulador no és mourà més d'un mm, per assegurar que no tinguem problemes de passar a través de parets.

Modificacions dyn_emu

- S'ha eliminat l'ús de semàfors per evitar problemes amb OSX.
  - Ús de dos queues thread-safe: q_rx i q_tx
- Les operacions sobre les cues s'han forçat a que no bloquegin el thread

```
if (is_rx_state) {
   if (queue_is_empty(&q_tx))
      continue;
} else {
   if (queue_is_full(&q_rx))
      continue;
}
```

- Inicialització i actualització del estat del simulador quan és requereixi:

```
void init_movement_simulator(const uint32_t *world);

void update_movement_simulator_values();
```

Sabeu que fa el const?

Elements del simulador

- Cada cert temps del sistema operatiu, actualitza el pas del simulador:

```
bool elapsed_time(clock_t t1, uint32_t miliseconds, int32_t *true_elapsed_time);
```

```
if (elapsed_time(t_last_upd, objective_delay,
&true_elapsed_time)) {
   objective_delay -= (true_elapsed_time - SIM_STEP_MS_TIME);
   t_last_upd = clock();
```

- Si ha passat el temps mínim del simulador, s'actualitza l'estat del robot:

```
calculate_new_position();
check_out_of_bounds();
update_sensor_data();
check_colision();
check_simulation_end();
```

El simulador s'executa un nombre finit de pasos. Modifiqueu el define segons la vostra necessitat.

Moviment del robot

- Les velocitats és llegeixen del mòdul sensor:

```
void _speed_dyn_2_speed_int(int16_t *v, uint8_t motor_id) {
   *v = dyn_mem[motor_id][DYN_REG__GOAL_SPEED_L];
   *v |= ((dyn_mem[motor_id][DYN_REG__GOAL_SPEED_H] & 0x03) << 8);
   if (dyn_mem[motor_id][DYN_REG__GOAL_SPEED_H] & 0x04) {
      *v *= -1;
   }
}
```

- La velocitat està escalada per una constant per limitar el desplaçament màxim a 1 mm/step del simulador

```
#define SIM_STEP_MS_TIME 50
#define DELTA_T ((float) SIM_STEP_MS_TIME/1000)
#define CNTS_2_MM ((float) 1.0/(DELTA_T*1023))
```

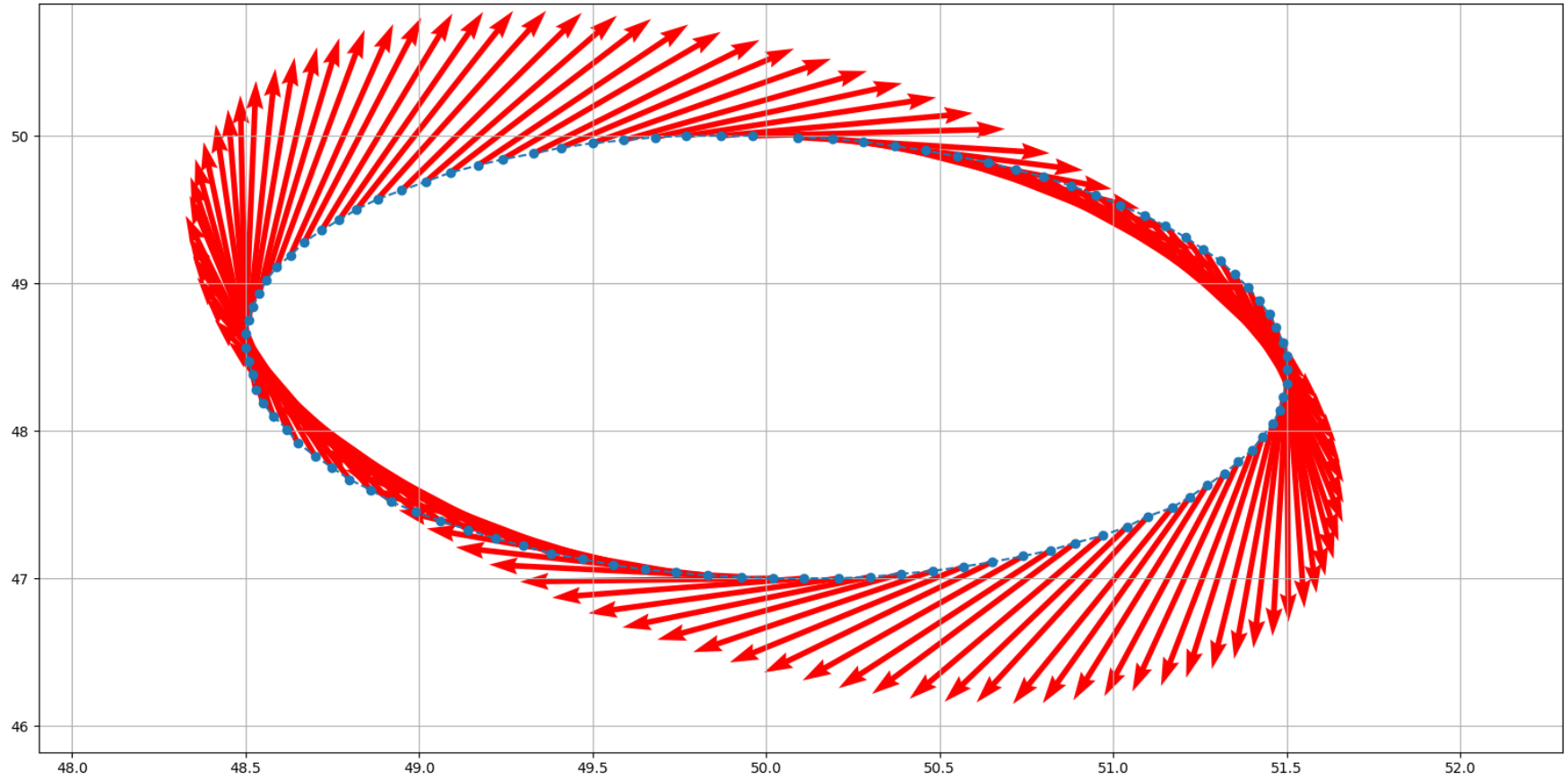
Moviment del robot

- Les equacions del moviment les podeu trobar explicades en aquest document si teniu interès: http://www.cs.columbia.edu/~allen/F15/NOTES/icckinematics.pdf
- Important! Necessitem (x,y) però també l'orientació:
  - 0º -> +x
  - 90º -> +y
  - +/- 180º -> -x
  - -90º -> -y
- Recordatori:
  - Mateixa velocitat en els dos motors: movement rectilini, endavant o endarrera depenent del sentit
  - Diferenta velocitat: gir en el sentit de la roda més lenta
  - Igual velocitat però signe contrari: pivotar en el mateix lloc

UNIVERSITAT DE BARCELONA

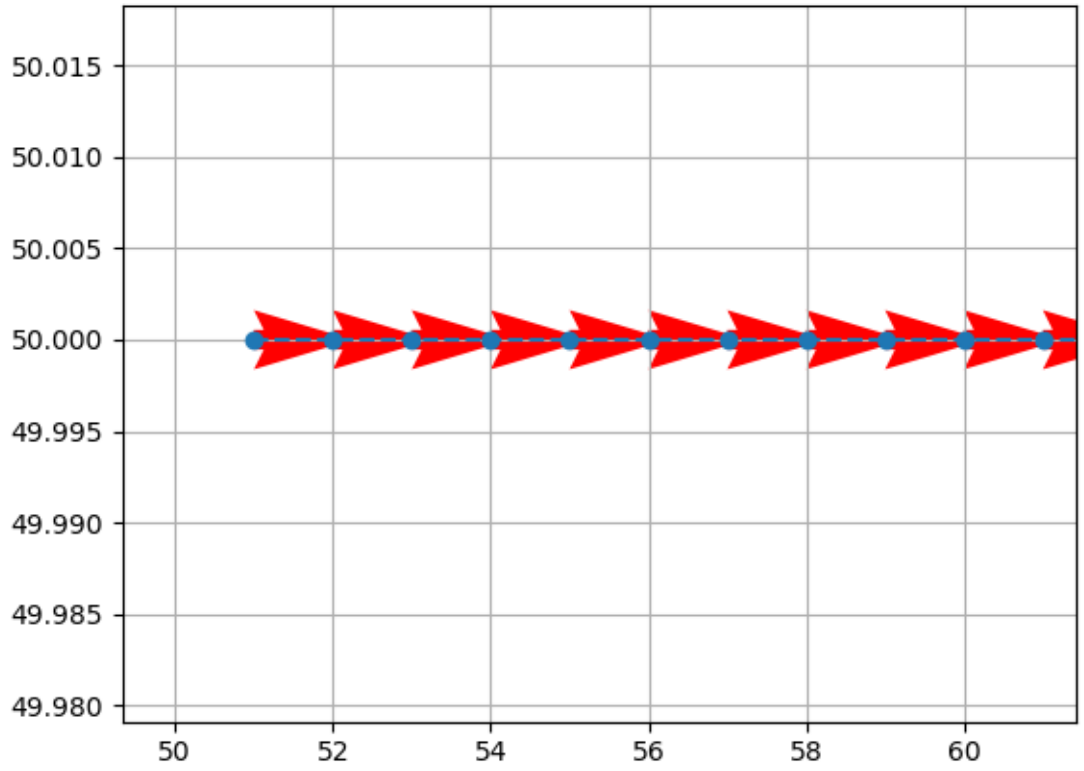
Moviment del robot

- Teniu un petit script en python (en evolució) per fer un plot de la trajectoria.
- Velocitats diferents:



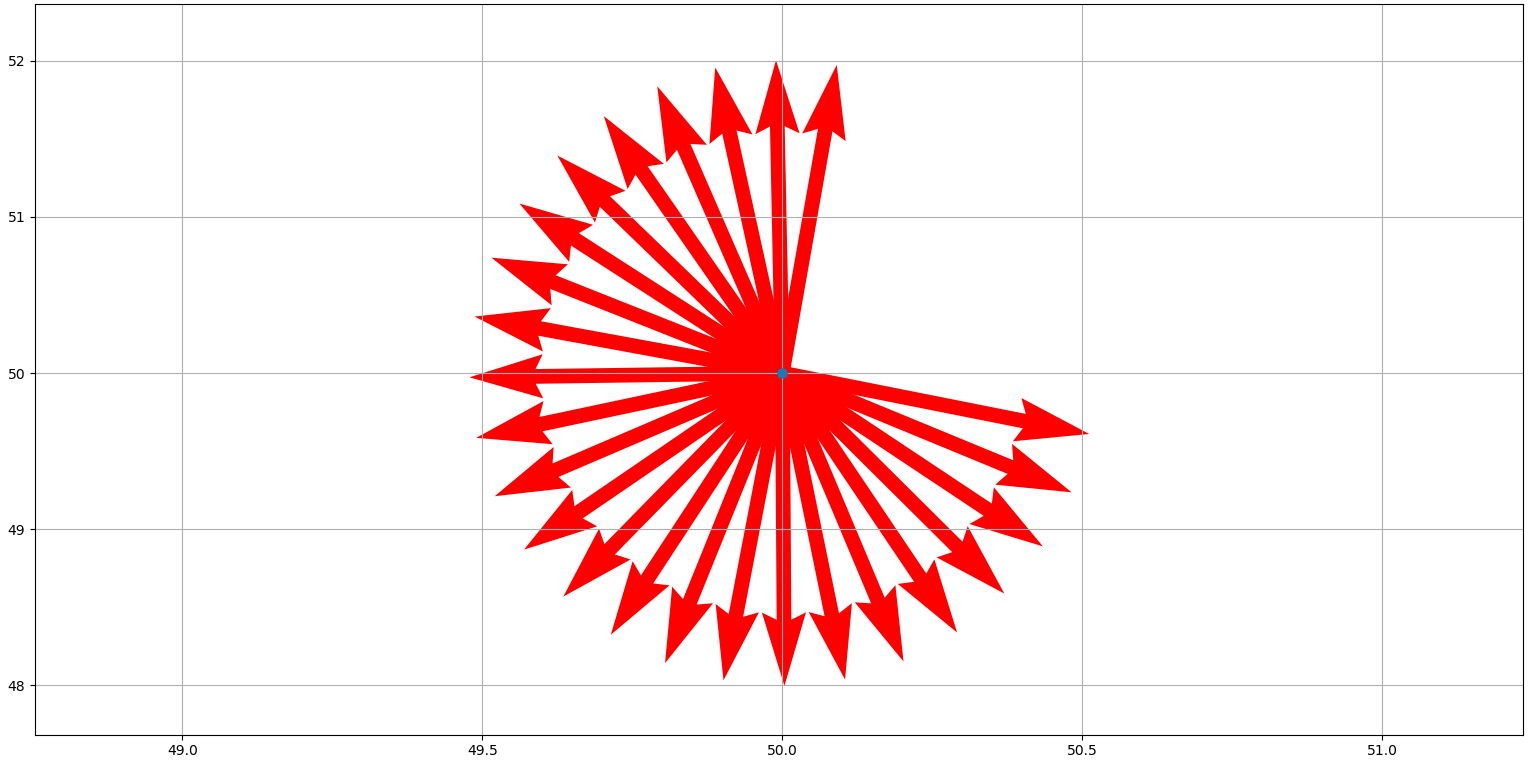
Moviment del robot

- Mateixa velocitat i sentit:



Moviment del robot

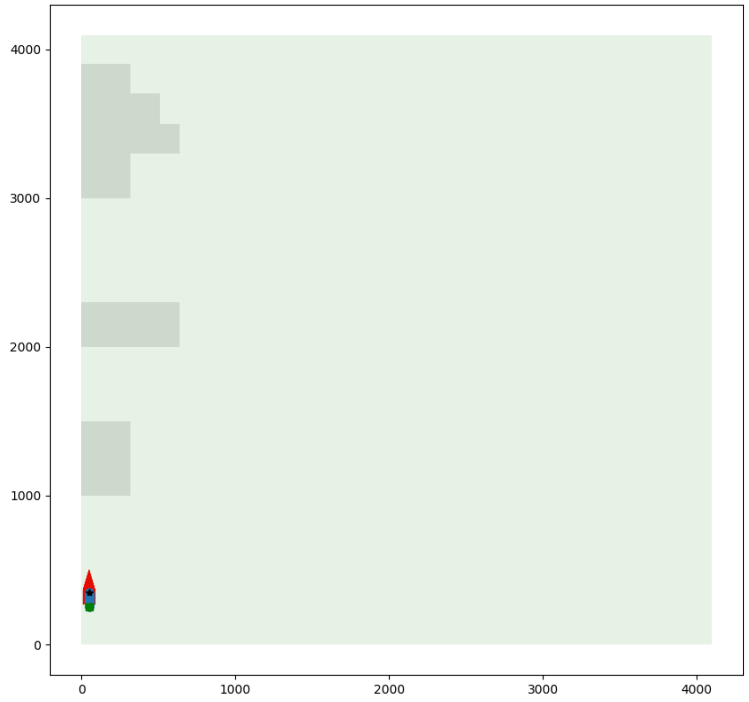
- Mateixa velocitat i sentit contrari:



Moviment del robot

- Seleccionant el nivell de debug adeqüat, és genera un log de la posició, velocitat i orientació.
- Per poder visualitzar posteriorment el recorregut, teniu un script de python que fa el plot d'aquests i l'habitació.

```
$ python plot_movement.py
```



Moviment del robot (cont)

- Comprovació no hem sortit de l'habitació:

```
void check_out_of_bounds() {
   if (robot_pos_str.x > ANCHO || robot_pos_str.y > ANCHO ) {
      printf("***** LEAVING ROOM... STOPPING SIMULATOR\n");
      fflush(stdout);
      end_simulator();
   }
}
```

- Sabeu per què no comprovem el cas negatiu? Pista: x,y són uint16_t
  - Que passa si li assignem a un unit16_t un valor negatiu?
- Actualització dades del sensor:

```
distance(&robot_pos_str, &dyn_mem[ID_SENSOR][DYN_REG__IR_LEFT],
      &dyn_mem[ID_SENSOR][DYN_REG__IR_CENTER],
      &dyn_mem[ID_SENSOR][DYN_REG__IR_RIGHT]);
```

Càlcul distància parets

- És fa el mateix pels tres sensors, però orientacions diferents:

```
x0 = robot_pos->x; //posicion del bloque de sensores = Posicion del robot
y0 = robot_pos->y; //posicion del bloque de sensores = Posicion del robot
theta = robot_pos->theta; //orientacion del sensor central, paralela a la del robot
theta_l = theta - M_PI / 2;
theta_r = theta + M_PI / 2;

//Sensor central:
sensor_distance(x0, y0, theta, robot_pos->world, centro, 1);
```

- La distància és calcula en la recta tenint en compte l'orientació del robot, arrodonint al enter més proper.

```
dx = cos(theta);
dy = sin(theta);
while ((modulo < 255) && !(obstaculo((uint16_t) (x0 + x), (uint16_t) (y0 + y), world))) {
   x += dx;
   y += dy;
   modulo = sqrt(x * x + y * y);
   indice++;
}
```

Dades habitació

- Un bit '0' indica no obstacle, '1' obstacle. Estructurat com un vector de uint32_t.

```
const uint32_t datos_habitacion [] = {
//Pared delante, en 0mm:
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, …
//1a zona sin obstaculo: (a 1000mm de distancia)
0x80000000, 0x00000000, 0x00000000, 0x00000000, …
```

- Cada 128 valors representa un salt a la següent fila, i.e., la distància entre el bit n de la paraula 0 i la paraula 127 és 1 en la direcció y.
- Les distàncies en 'x' en canvi tenen en compte la posició dels bits individuals. E.g. en el cas 0x80000001 hi ha una distància de 30 mm i en el cas de 0x00000001, 0x80000000 hi ha una distància de 1 mm.
- Es proporcionen funcions per treballar més fàcilment amb aquestes dades.

Conversió coordenades a elements vector

- La funció coordenades permet fer la conversió de (x,y) a element del vector i bit:

```
void coordenadas(uint16_t x, uint16_t y, uint32_t *p_offset, uint8_t *p_bit) {
   uint32_t p = 0;
   uint8_t b = 0;
   //En que fila me encuentro?
   //Cada incremento de "y", corresponde a un salto de 1 fila de 128 bloques
   p = y << 7;    //INT(y*128)
   //Dentro de la fila, nos hemos movido hasta el bloque INT(x/32):
   p += x >> 5;
   //Lo que queda de esta division nos da el bit dentro del bloque:
   //31 - x - INT(x/32)*32
   b = (32 - 1) - (x - ((x >> 5) << 5));

   *p_offset = p;
   *p_bit = b;
}
```

Detecció de paret

- La funció obstaculo ens permet saber directament si en la posició (x,y) hi ha un paret:

```
uint8_t obstaculo(uint16_t x, uint16_t y, const uint32_t *mundo) {
   uint32_t offset = 0;
   uint8_t bit = 0;
   coordenadas(x, y, &offset, &bit);
   if ((mundo[offset] & (1 << bit)))
      return true;
   return false;
}
```

Avaluació projecte

- S'ha d'entregar:
  - Codi
  - Memòria del projecte
  - Vídeo de teleconferència amb una presentació de 10 minuts del projecte

- Els professors revisaran el material i més endavant hi haurà unes franges curtes i preestablertes per cada grup en les quals és faran preguntes sobre el projecte entregat.

Sessions pràctiques

- Durant 1h de les sessions de pràctiques els professors estaran connectats per si voleu resoldre dubtes:
  - Grup dilluns: 13h a 14h
  - Grup dimarts: 8h a 9h
  - Grup dijous: 9:30 – 10:30
  - Grup divendres: TBD
- Podeu continuar sol·licitant informació o teleconferències fora d'aquestes hores per correu electrònic als professors.

- Migració a Clion:
  - https://www.jetbrains.com/clion/download/
  - https://www.jetbrains.com/es-es/community/education/#students
  - MSYS2: pacman -S mingw64/mingw-w64-x86_64-cmake mingw64/mingw-w64-x86_64-make