Hindawi
Journal of Robotics
Volume 2017, Article ID 8069654, 9 pages
https://doi.org/10.1155/2017/8069654

Research Article

ARM-Cortex M3-Based Two-Wheel Robot for Assessing Grid Cell Model of Medial Entorhinal Cortex: Progress towards Building Robots with Biologically Inspired Navigation-Cognitive Maps

J. Cuneo, L. Barboni, N. Blanco, M. del Castillo, and J. Quagliotti

Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Julio Herrera y Reissig 565, 1300 Montevideo, Uruguay

Correspondence should be addressed to L. Barboni; lbarboni@fing.edu.uy

Received 25 November 2016; Revised 20 February 2017; Accepted 12 March 2017; Published 9 April 2017

Academic Editor: Liwei Shi

This article presents the implementation and use of a two-wheel autonomous robot and its effectiveness as a tool for studying the recently discovered use of grid cells as part of mammalian's brains space-mapping circuitry (specifically the medial entorhinal cortex). A proposed discrete-time algorithm that emulates the medial entorhinal cortex is programed into the robot. The robot freely explores a limited laboratory area in the manner of a rat or mouse and reports information to a PC, thus enabling research without the use of live individuals. Position coordinate neural maps are achieved as mathematically predicted although for a reduced number of implemented neurons (i.e., 200 neurons). However, this type of computational embedded system (robot's microcontroller) is found to be insufficient for simulating huge numbers of neurons in real time (as in the medial entorhinal cortex). It is considered that the results of this work provide an insight into achieving an enhanced embedded systems design for emulating and understanding mathematical neural network models to be used as biologically inspired navigation system for robots.

1. Introduction

This article describes the design and use of an ARM-Cortex M3-based board two-wheel robot in which the neurons within the space-mapping system are named *grid cells* (GCs hereafter). The GCs were discovered in 2005 by M.-B. Moser and E. I. Moser, who received the Nobel Prize for Medicine in 2014 [1]. The GCs have periodic triangular arrays and are hypothesized to function as the mammalian (e.g., rats and mice, possibly also humans) positioning-metric navigation system [2–4]. They are place-modulated neurons positioned on the medial entorhinal cortex (MEC hereafter). Multiple firing locations are found on these triangular arrays when an individual moves around a particular location. The firing information is then posted towards more structured neural networks to control movements (for instance, to *place cells*) [5]. In this respect, research about GCs is addressed using extracellular action potentials recorded from MEC neurons by means of an array of electrodes placed on freely moving rodents.

Neurobiologically based robot navigation has been the subject of research for a considerable time as noted in [6]. However, recent scientific interest has been focused on understanding the mechanisms by which these GCs create a cognitive map of the environment as this would enable the building of robots that mimic the mammalian biological navigation system.

Previous innovative studies are recognized with respect to this achievement. For example, [7, 8] show a new architecture for generating GCs that was implemented into a real robot. These GCs are based on various modulo operators applied on the path integration field. However, there are drawbacks with this approach as it has a path integration error problem and the robot has a width and height of 40 cm and 1 m, respectively (the robot is the Robulab 10 from Robosoft [9]). Another example: the study by [10] implemented a GC model

Table 1: Scaled variables and parameters values used for equations (3), (4), and (5).

$v_i' = 10^3 v_i$	$v_r' = 10^3 v_r$	$u_{i-1}' = 10^3 u_{i-1}$
$v_{i-1}' = 10^3 v_{i-1}$	$I' = 10^3 I$	$I' = 10^3 I$
$v_{\text{off}}' = 10^3 v_{\text{off}}$	$u_i' = 10^3 u_i$	$a' = 10^3 a$
$v_{\text{th}}' = 10^3 v_{\text{th}}$	$v_r = -60\,\text{mV}$	$c = -50\,\text{mV}$
$a = 0.03\,\text{s}^{-1}$	$v_{\text{th}} = -40\,\text{mV}$	$d = 100\,\text{pA}$
$b = -2\,\text{n}\Omega^{-1}$	$C = 10\,\text{nF}$	$v_{\text{peak}} = 35\,\text{mV}$
$k = 0.7\,\mu\text{A/V}^2$	$v_{\text{off}} = -v_r$	$I_{\max} = 0.17\,\text{pA}$

based on a continuous attractor network of rate-coded cells, which encode the robot location and orientation (this robot is the Pioneer 2DXe mobile robot from Mobile Robots Inc. [11]).

In this work, we implement a novel discrete-time differential equation GC neuron model [12] onto a 32-bit ARM-Arduino Due based hardware-software platform. This work contributes to assessing the mathematically tractable and programmable model proposed by [12] and a set of suitable parameters are obtained for use (interested readers can see in advance (3), (4), (5), and Table 1).

On the other hand, previous works use commercial robot platforms. Instead, in this work the robot has nongeneralizable design; that is, the GC mathematical model requirements and hardware features were taken into account to accomplish the best performance. This robot is not intended to be able to be adapted to other purposes other than an effective configurable tool for studying GCs and MEC models and their dynamics.

In developing the robot, the following was achieved: (i) a suitable algorithm was obtained for the discrete-time model by using an Arduino hardware-software platform; (ii) input signals from the environment were used as input for the GCs; and (iii) the firing patterns generated by the algorithms were determined while the robot is freely moving. However, clear drawbacks were determined at a hardware level while achieving the above, and it was determined that only a reduced number of GCs can be implemented. Although this issue needs to be addressed in future work to overcome this hardware limitation and achieve a huge GCs network, this study demonstrates the mathematically tractable GC model proposed in [12] is programmable on RISC microcontrollers. In addition, parameters' values set was found for which the GCs worked as predicted.

This study therefore offers two benefits: (i) an assessment of an autonomous robot platform that mimics part of the mammalian biological navigation system (for this reason a PC is not used to simulate GCs, as the target is to develop robots using biologically inspired navigation-cognitive maps); (ii) ensuring that the robot is an effective and configurable tool for studying and validating real biological navigation scenarios and that it enables experimental repeatability (in contrast to relying on the free and random moving of rodents). In other words, this robot is conceived to be a research tool for use by neuroscientists to study how the MEC of mammalians builds an environmental cognitive-neural map.

This paper is presented as follows: Section 2 describes the mathematical model of the GCs. Section 3 presents implementation of GCs within the robot; Section 4 describes the mathematical model translation to code by scaling the discrete-time differential equations. Section 5 shows firing of the GCs which is achieved when the robot explores certain locations within the laboratory arena, and final conclusions are made in Section 6.

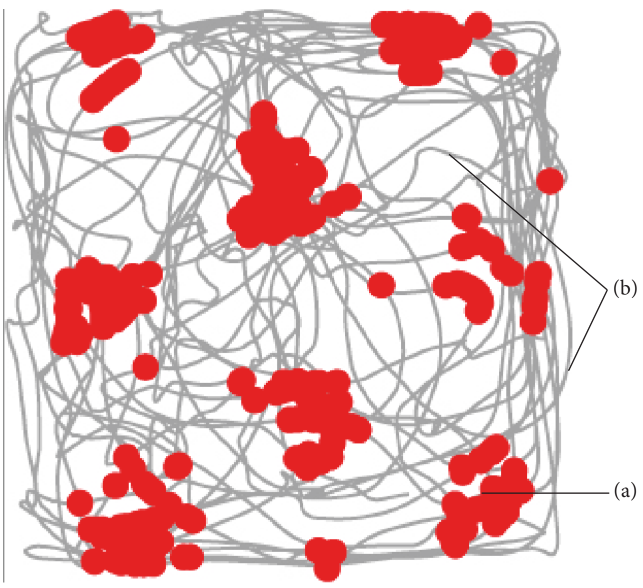

Figure 1: *Grid cells* in the MEC of a rat while moving in an arena (from [5]): (a) set of spikes (in red) that appears while the rat is moving around certain locations, (b) random walk trajectories in black.

2. *Grid Cell* (GC) Neural Model

The GCs comprise the medial entorhinal cortex (MEC) of mammalian species. These cells provide the brain with a metric for the animal's local space (or geometric reference) so that it knows its position in relation to its environment; that is, they work as a cognitive map with firing fields dispersed over the entire environment. For example, when a mouse moves in a given space, the GCs show shots in a hexagonal firing pattern with equidistant nodes, as shown in Figure 1.

As a model, this work uses (1), as proposed in [12, 13] and takes into account three dimensions of GC variation: scale, orientation, and phase. It is of note that the GCs model is too broad to be captured completely in this article, and thus only the base ideas are introduced to enable an understanding of

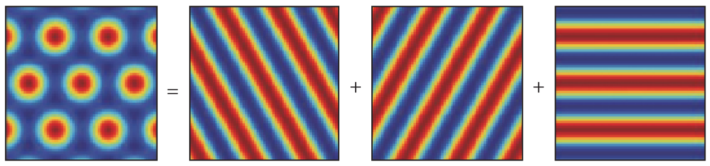

FIGURE 2: *Grid cell* $g_n(x, y)$ function as the sum of three two-dimensional 0, 60, and 120 degrees out of phase sinusoids (from [13]).

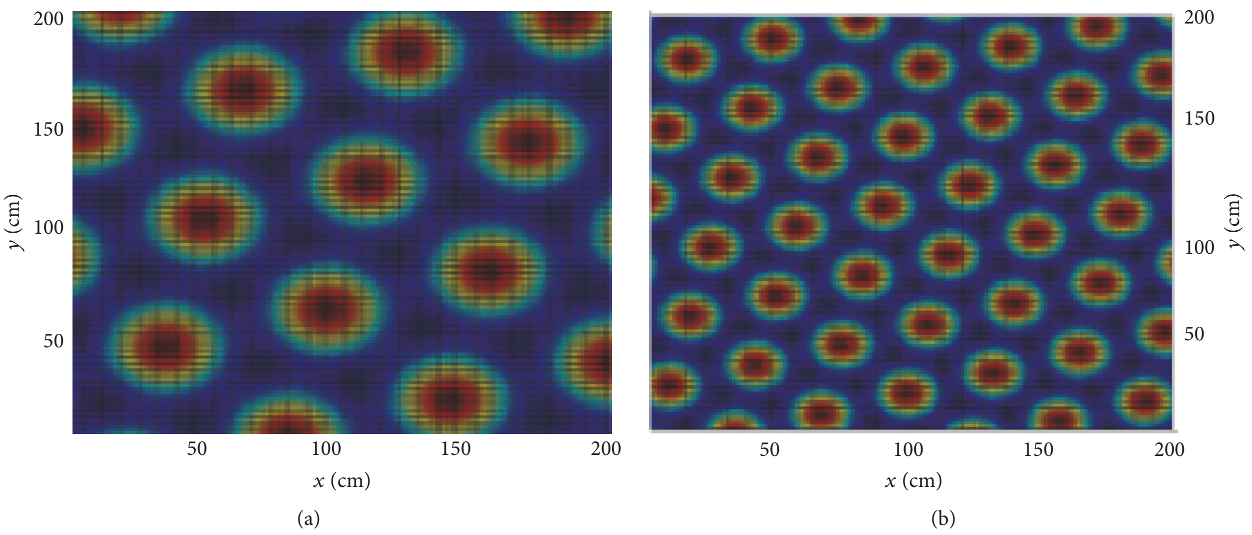

FIGURE 3: Examples of GCs with different grid spacing and field size activities (a) $g_n(x, y)$ function with $\lambda_n = 1.6$ m (coarse-scale grid) and (b) with $\lambda_n = 2.9$ m (fine-scale grid). It is of note how the GCs build a coordinate map composed of multiple grid layers, each with a different length scale between grid points, but with fixed spacing between grid points for each nth GC. By combining multiple-grained grid layers it is possible to enable an Euclidean coordinate map and identify point of interest.

how the GC and the robot work. nth GC $g_n(x, y)$ function is the sum of three two-dimensional sinusoids and is characterized by its wave vector k_n, with 0, 60, and 120 degrees of angular difference (see (1)). The index n indicates the value variations of GC parameters that depend on the their position inside the MEC. The expression is given by (1); Figures 2 and 3 give examples of possible shapes, to provide an enhanced insight.

$$g_n(r) = \frac{2}{3} g_n^{\max} \left[\frac{1}{3} \sum_{j=1}^{3} \cos\left(k_{j,n}(r - r_0)\right) + \frac{1}{2} \right], \quad (1)$$

where $r = (x, y)$ is the position vector, $r_{0,n} = (x_{0,n}, y_{0,n})$ is a referential point, $g_n^{\max}$ is a parameter, and λ_n is the number of nodes per unit distance related to the wave vector k_n as $\lambda_n = 4\pi/\sqrt{3}k_n$. The wave vector in (1) is as follows:

$$\begin{aligned} k_{n,j} = \frac{k_n}{\sqrt{2}} \Bigg[&\cos\left(\theta_n + \frac{q_j \pi}{12}\right) \\ &+ \sin\left(\theta_n + \frac{q_j \pi}{12}\right), \cos\left(\theta_n + \frac{q_j \pi}{12}\right) \\ &+ \sin\left(\theta_n - \frac{q_j \pi}{12}\right)\Bigg], \end{aligned} \quad (2)$$

where $q_j = 1$ for $j = 1$, $q_j = 5$ for $j = 2$, and $q_j = 3$ for $j = 3$ and $\|k_{n,1}\| = \|k_{n,2}\| = \|k_{n,3}\| = k_n$. It is used to express the GCs membrane potential as (3), (4), and (5).

In addition the $g_n(r)$ function uses the following parameters (depending on the GC position inside the MEC):

(i) λ_n [grids/m] $= 3 - 1.5\chi_n$
(ii) χ_n value as n/N, where N is the number of GC neurons
(iii) $\mathcal{N}(0, 1)$ which is a random value normal distribution
(iv) θ_{χ_n} [rads] $= 2\pi\mathcal{N}(0, 1)$
(v) θ_{env} [rads] angular phase shift that the environment produces on the grid
(vi) θ_n [rads] $= \theta_{\text{env}} + \theta_{\chi_n}$
(vii) $r_{0,n} = (2\mathcal{N}(0, 1), 2\mathcal{N}(0, 1))$ (horizontal and vertical offset of the cell): factor 2 is used because the robot will be used in an 2×2 m testing area

The fire time-discrete nonlinear differential equation is proposed in [14], which describes the GC membrane potential; it is also implemented. The assessed model of nth GC is then expressed by (3), (4), and (5) with the following parameters: (i) $v_{n,i}$ is the membrane potential (unit mV) at the ith time-step; (ii) $u_{n,i}$ is the recovering potential (unit mV) at the ith time-step; (iii) t is the ith time-step value; that is, $t = t_{i+1} - t_i$

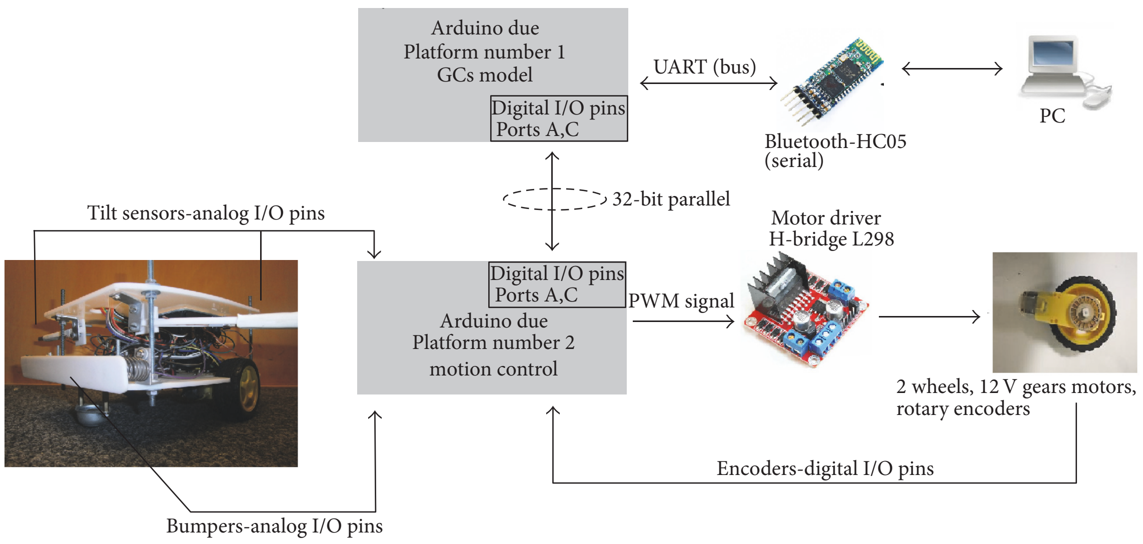

FIGURE 4: Hardware block diagram of robot.

(discrete-time value for numerical integration); (iv) $I_{n,i}$ is the synaptic excitation current model from $g_n(r, i)$ at the ith time-step; (v) $v_{n,\text{off}}$ is the shifting value to enable all positive values to be achieved (for easy digital implementation), (vi) $v_{n,\text{peak}}$ is the maximum achievable membrane potential after firing, (vii) $v_{n,r}$ is the refractory membrane potential (unit mV), (viii) $v_{n,\text{th}}$ is the firing threshold (unit mV), (ix) c is the lowest recovery membrane after firing (unit mV), and finally (x) a, b, C, and d are constants (interested readers can see in advance the model results obtained after the assessment in Table 1).

$$\begin{aligned} v_{n,i} = {} & v_{n,i-1} \\ & + \frac{t}{C}\left[k\left(v_{n,i-1} - v_{n,\text{off}} - v_{n,r}\right)\left(v_{n,i-1} - v_{n,\text{off}} - v_{n,\text{th}}\right)\right. \\ & \left. + I_{n,i} - u_{n,i-1} + b\left(v_{n,\text{off}} + v_{n,r}\right)\right], \end{aligned} \tag{3}$$

$$u_{n,i} = u_{n,i-1} + a\left(bv_{n,i-1} - u_{n,i-1}\right), \tag{4}$$

$$I_{n,i} = I_{\max} g_n\left(r_i\right). \tag{5}$$

After firing, the membrane potential v_n (mV) and the recovering potential $u_{n,i}$ (unit mV) need to then be reset according to the following rule: if $v_n > v_{n,\text{peak}} + v_{n,\text{off}}$ then $v_{n,i} = c + v_{n,\text{off}}$ and $u_{n,i} = u_{n,i} + d$.

3. Implementation of Robot Hardware

The robot system block diagram (shown in Figure 4) features two microcontrollers interconnected by a tailored 32-bit parallel wired communication using IO ports A and C. One of the microcontrollers updates is the robot position relative to the starting location over time (odometry) from information obtained from the wheels' encoders. The $r = (x, y)$ position vector is sent to the other microcontroller, which solves discrete-time equations (3), (4), and (5). Both the embedded systems are Arduino Due boards [15] based on the 32-bit SAM3X8E ARM core Cortex-M3 architecture. Each board features a 84 MHz clock, 54 digital input/output pins, and 12 analog inputs and the robot's wheels are driven by two 12 V DC motors with PWM H-bridge driver. The robot communicates to the outside using the PC, so that the GC spikes and the robot's position can be read by the researchers. The robot constitutive parts are shown in Figures 5 and 6. The axles of its wheels have slotted disks to track rotation using optocouplers (the set disk-optocoupler is called the "encoder" and is used to estimate the wheel's angular rotation). The robot estimates its displacement and trajectory from the departure point using the information provided by such encoders (odometry) as suggested in literature [16] (pp. 49–54). The position $r = (x, y)$ is estimated with an accuracy of 1 cm, which is sufficient for this first GC model assessment. The tilt sensors are resistors whose values vary with deformation. They are used to detect the proximity of obstacles and avoid collisions. Figure 6 shows a collision.

The bumper has the same function but is connected to two lateral switches that have state change of ON/OFF which indicate possible frontal collisions with an object. The robot is power-supplied by an external unified DC voltage power supply of 12 V, 5 V, and 3.3 V. In addition the robot includes a camera OV5642 [17] for taking RGB photos, which will enable it in future research to identify different environments; however at this assessment level, the environment for the GC model is only provided in terms of the constant parameters as described. The robot also includes a Bluetooth HC-05 communication module [18] for code debugging messages during the early development stages as discussed in Section 5. This module enables communication to be established with the PC to obtain real time GCs spikes.

4. Implementation of Robot Software

The Arduino IDE [15] was used as the programming environment as it provides libraries built in C++ and thus reduces the time required to obtain a code prototype. The programs

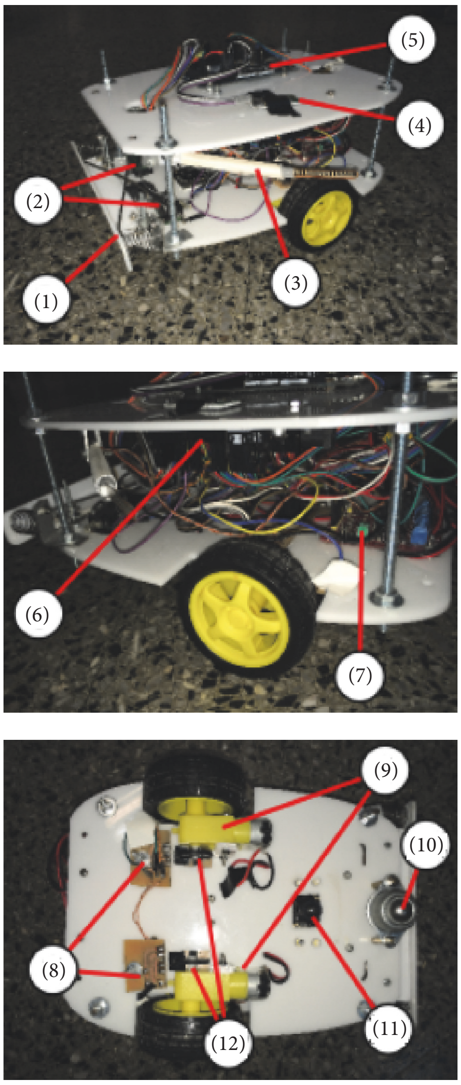

Figure 5: Constitutive parts of robot: (1) bumper, (2) bumper sensors (switches), (3) tilt sensor, (4) Bluetooth module, (5) Arduino Due board number 1 (this implements the GCs, i.e., (3), (4), and (5)), (6) Arduino Due board number 2 (this implements the trajectory tracking as proposed in [16]), (7) motor driver L298N [19], (8) photodetector, (9) wheels and motors (gears are fixed to wheels), (10) small freewheel, (11) color camera TTL OV5642 [17], and (12) slotted disk.

in each microcontroller were implemented using Round-Robin scheduling with interruptions; this preemptive process scheduling algorithm was selected because of its ability to handle the multiple sensors signals. In addition, it has the computational requirements to obtain numerical solutions of the GCs. The scheduling pseudocode of the Round-Robin for the navigation system is shown in Listing 1 to clarify its operation. The robot's motion occurs using a uniform random distribution function that generates random wheels motion (speed and direction for each wheel: both are updated independently periodically each 1 s). The motion simulates that of a rat exploring certain environments and while exploring the robot estimates its position relative to the starting point using the information provided by encoders (odometry-based model [16]). During this assessment stage, the error sources such as unequal wheel diameters, variation in the contact point of the wheels, and variable friction are neglected. A programing subroutine that we named RoutineAlert() was implemented using Arduino IDE libraries, which provide randomly generated values. The following two problems were then analyzed: (i) the need to determine limited speeds values achievable by each wheel gear to obtain bounded speed values and (ii) the need to radically reduce the total time consumed by the processor to perform the necessary tasks between consecutive RoutineAlert() calls. An important concept in dealing with the design is that the randomized values should not occur frequently (i.e., one RoutineAlert() calls inside the Round-Robin) as the robot speed would then tend to average the distribution and randomness would be lost.

4.1. Odometry and Robot Walk-Time: Avoiding Position Errors of Significance.

The odometry system performs tracking of the robot's position using the rotational angle of the wheels. The model used was proposed in [16] and is widely used for two-wheeled robots with differential traction. It uses the difference between wheel speeds to determine the direction of movement; as the wheels have a fixed axis, walking is achieved by imposing different speeds to each engine. The algorithm simply models the system and defines an axis of coordinates in the plane. The model is shown in Figure 7 and the equations used to update the robot position (x, y) are described below:

$$\begin{aligned} x_i &= x_{i-1} + \Delta S_i \cos(\theta_i), \\ y_i &= y_{i-1} + \Delta S_i \sin(\theta_i), \\ \theta_i &= \theta_{i-1} + \Delta\theta_i, \\ \Delta\theta_i &= C_m \frac{N_{R_i} - N_{L_i}}{L}, \\ \Delta S_i &= C_m \frac{N_{R_i} + N_{L_i}}{2}, \\ C_m &= \frac{2\pi R}{C_e}, \end{aligned} \tag{6}$$

where C_m is the ratio of the number of pulses to the distance traveled (meter per encoder pulse), N_{R_i} and N_{L_i} are the amount of measured pulses on the right and left wheel up to time t_i, L is the distance between the wheels' centers, R is the wheel radius, and C_e is the encoder resolution (counts per turn). Cumulative errors occur when updating the robot's position and the article [20] addresses the problem of measurements and correcting errors in systems such as this. According to these authors, among the multiple sources of error that could affect a system with two controlled wheels and odometry, two are more dominant than the others: (i) (systematic error) the radii of both wheels are not equal (wheels are coated with rubber to improve traction and it is very difficult to ensure that they are manufactured to be exactly the same); thus an encoder pulse can mean that different linear displacements occur with respect to

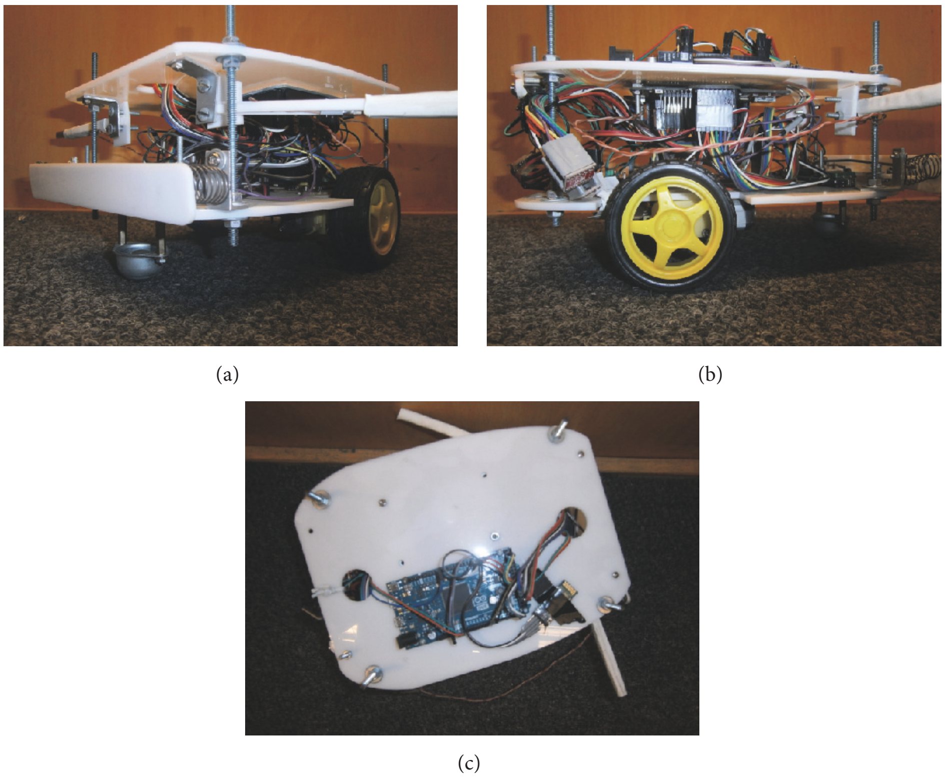

Figure 6: Constitutive parts of robot: (a, b) left and right sides views; (c) top view of the robot: observe the left tilt sensor that indicates wall collision.

```

while (true){
    if (right wheel rotates){
        number of encoder pulses increases;
    }
    if (left wheel rotates){
        number of encoder pulses increases;
    }
    if (sensor interruption flag indicates collision){
        execute collision solver, call RoutineAlert();
    }
    if (there is not collision){
        robot walking continues, call RoutineAlert();
        execute new (x,y) position estimation;
        (x,y) to second microcontroller to update GCs states;
    }}
```

Listing 1: Round-Robin scheduling pseudocode to manage navigation.

each wheel and the assumed straight trajectory that should occur in the odometry model is thus actually curved; (ii) (random error) wheel slippage: this causes a change in the relation between the linear displacement of the wheel and the measured angle of rotation by the encoders. This source generates similar effects to those mentioned above but in rectilinear trajectories and such effects are mainly significant at angles of rotation. These sources of error modify the robot's actual position in relation to that of its estimated position. Although positional compensation has not been implemented in this preliminary work to avoid errors, it has been empirically found to be possible to avoid such error compensation as suggested in [20] and that an assessment of GCs is sufficiently accurate if the robot's walk-time is less than 5 min. Therefore, the odometry system has a limited *useful life* for each robot walk and for the surface in which it moves and the value of 5 min is used in this considered experimental setup. Each robot walk-time was thus only 5 min (we also

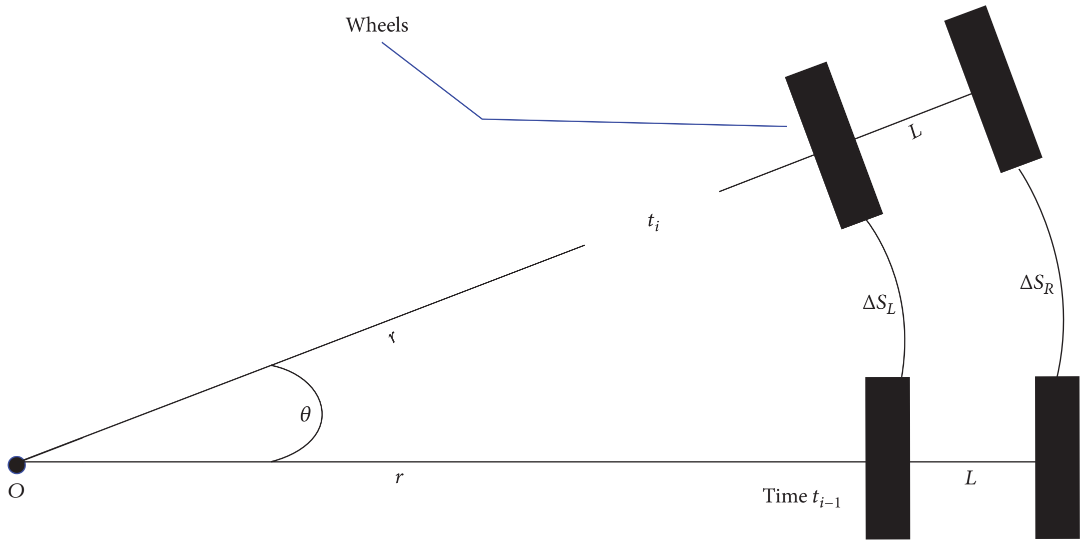

FIGURE 7: Odometry navigation model.

reiterate this value in the results section to explain results shown in Figure 8). The starting point is defined inside the arena prior to the robot walking. The updated position is then used to compute each of GCs' states (i.e., to solve (3), (4), and (5)). In case of collision with obstacles, the robot randomly updates speed and rotates its wheels at different speeds to enable it to change course.

4.2. Scaling Equations. One of the main issues requiring attention in software implementation is the need to ensure that the computation of each state for each time-step is very short. In this respect, using the selected Cortex M3 it is possible to solve (3), (4), and (5) with a reduced number of clock cycles (i.e., a small time-step). Therefore, it is necessary to solve them using integer values, and to achieve this, the equations need to be scaled because the processor uses its hardware multiplier module. This module enables product operation in just a few clock cycles, thereby reducing the computational time, and also by scaling, the overall computation time thus is reduced per GC from 40 μS to 5 μS.

Scaling was performed ad hoc and according to the following reasoning (the function that describes the voltage membrane behavior is used here as an example); the variable v_i was then multiplied by a factor in a way that was acceptable for truncation to integer values with respect to the error due to the floating point usage. A factor of 1000 achieves this condition (values 1000 times lower than the dimension on v_i can be neglected). The scaled variables were then used for (3), (4), and (5) as shown in Table 1. This conclusion is extremely important for use as principle or rule in this class of bioinspired robot design. To give better insight, by scaling (3) it becomes

$$\begin{aligned}10^3 v_{n,i} &= 10^3 v_{n,i-1} \\ &\quad + \frac{t}{C}\left[10k\left(10^3 v_{n,i-1} - 10^3 v_{n,\mathrm{off}} - 10^3 v_{n,r}\right)\right. \\ &\quad \cdot \left(10^3 v_{n,i-1} - 10^3 v_{n,\mathrm{off}} - 10^3 v_{n,\mathrm{th}}\right) + 10^3 I_{n,i} \\ &\quad \left. - 10^3 u_{n,i-1} + b\left(10^3 v_{n,\mathrm{off}} + 10^3 v_{n,r}\right)\right].\end{aligned} \tag{7}$$

Thus, the scaled equation with new names of variables that the program into the microcontroller resolves is the following:

$$\begin{aligned}v'_{n,i} &= v'_{n,i-1} + \frac{t}{C}\left[10k\left(v'_{n,i-1} - v'_{n,\mathrm{off}} - v'_{n,r}\right)\right. \\ &\quad \cdot \left(v'_{n,i-1} - v'_{n,\mathrm{off}} - v'_{n,\mathrm{th}}\right) + I'_{n,i} - u'_{n,i-1} \\ &\quad \left. + b\left(v'_{n,\mathrm{off}} + v'_{n,r}\right)\right].\end{aligned} \tag{8}$$

5. Results of Assessment

A testing stage was conducted to evaluate the platform performance and its effectiveness in assessing GCs models. This was achieved by computing a maximum of 200 GCs in a time-step of 1 ms and reporting the spikes events to the PC using the mentioned Bluetooth module. The transmitted data frame format is shown in (9), in which each field is a byte. Data were processed using a Python written script and the GCs spikes and robot trajectories are plotted as shown in Figure 8.

$$\begin{gathered}x_0, y_0, \mathrm{spike}_{\mathrm{GC}_1}, \mathrm{spike}_{\mathrm{GC}_2}, \ldots, \mathrm{spike}_{\mathrm{GC}_n}, \\ \vdots \\ x_m, y_m, \mathrm{spike}_{\mathrm{GC}_1}, \mathrm{spike}_{\mathrm{GC}_2}, \ldots, \mathrm{spike}_{\mathrm{GC}_n}.\end{gathered} \tag{9}$$

It was important to ensure that the number of GCs spikes sent to the PC for analysis is $n = 56$ in (9): this number was empirically determined as the maximum number of GCs spikes and positions that could be sent every 1 ms at

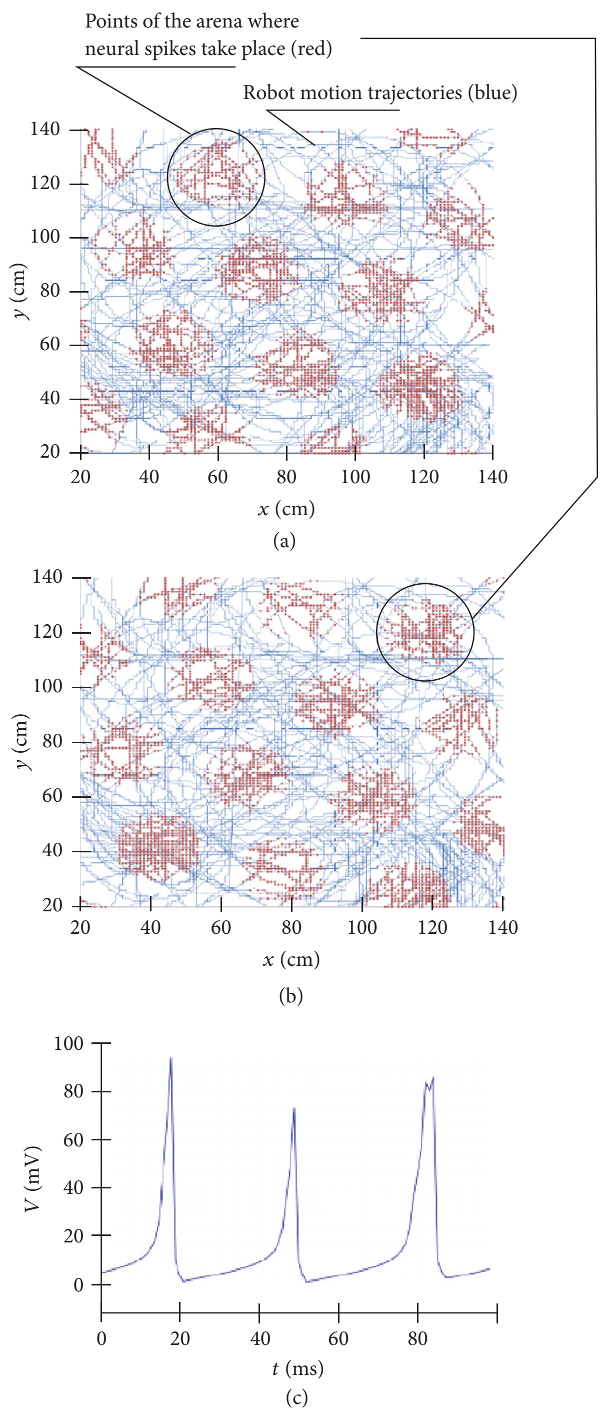

Figure 8: (a) Experimental map showing firing rate distribution for same *grid cell* with $\lambda = 2.69\,\mathrm{m}^{-1}$ (spike locations of *grid cell* are superimposed on robot's trajectory), (b) with $\lambda = 2.74\,\mathrm{m}^{-1}$; (c) an example of numerical integration of *grid cell* potential-membrane spikes (see (3)) for a given static robot position. Please, observe the triangular array of spikes.

115200 BPS without errors. When the Arduino library is used to access a Bluetooth Serial port it may produce an overhead and the implemented Round-Robin software architecture with interruptions is then not able to properly handle the information at the required time. For example, data have been lost when $n > 56$. For the data obtained using a set of suitable parameters values for (3), (4), and (5) (please see Table 1) a map can be found of the firing distribution of each GC for every location within the explored area in addition to the appearance of firing frequency. With the model proposed in [12] ((3), (4), and (5)), a unified and mathematically tractable and programmable model was achieved (but still partially tested) and further research can now continue. To generate the case study shown in this work, several repetitions of the above-mentioned procedure were conducted and files of the format given by (9) were obtained when the robot was placed in a laboratory arena, from a starting point chosen as $(x_0, y_0) = (40, 40)$ cm and $\theta_0 = 0$. The robot then began to move randomly, while simultaneously sending generated data to the PC. The procedure ended at 5 min after it began and Figure 8 shows superposition of three experiments (each with a duration of 5 min) for a given GC with two different λ values. Table 1 shows the suitable scaled variables and the suitable parameters values used for (3), (4), and (5).

6. Conclusion

This article shows the use of a two-wheel autonomous robot and confirms its effectiveness as a configurable tool for studying GCs models. A computational maximum of 200 GCs was achieved with a time-step of 1 ms; however this number of GCs is not yet representative of an actual mammalian MEC and thus more computational resources will be required in future developments. The main limitation of this study is related to data communication, and it will thus be necessary to use high speed wireless or wired communications, probably embedded systems with Ethernet communications, such as the well-known Raspberry Pi board. The assessment shows that the GC model works as predicted and a mathematically tractable and programmable model with parameters set was obtained.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding publication of this paper.

References

[1] M.-B. Moser and E. I. Moser, "The 2014 Nobel Prize in Physiology or Medicine," https://www.nobelprize.org/nobel_prizes/medicine/laureates/2014/press.html.

[2] E. I. Moser, Y. Roudi, M. P. Witter, C. Kentros, T. Bonhoeffer, and M.-B. Moser, "Grid cells and cortical representation," *Nature Reviews Neuroscience*, vol. 15, no. 7, pp. 466–481, 2014.

[3] K. M. Igarashi, "The entorhinal map of space," *Brain Research*, vol. 1637, pp. 177–187, 2016.

[4] C. Barry and D. Bush, "From A to Z: a potential role for grid cells in spatial navigation," *Neural Systems & Circuits*, vol. 2, no. 1, article 6, 2012.

[5] C. Palmer, "Electrical Oscillations Found to be Critical for Storing Spatial Memories in Brain," Regents of the University of California, http://www.sciencenewsline.com/news/2011042913000039.html.

[6] P. J. Zeno, S. Patel, and T. M. Sobh, "Review of neurobiologically based mobile robot navigation system research performed since 2000," *Journal of Robotics*, vol. 2016, Article ID 8637251, 17 pages, 2016.

[7] A. Jauffret, N. Cuperlier, and P. Gaussier, "From grid cells and visual place cells to multimodal place cell: a new robotic architecture," *Frontiers in Neurorobotics*, vol. 9, article 1, 2015.

[8] A. Jauffret, N. Cuperlier, P. Gaussier, and P. Tarroux, "Multimodal integration of visual place cells and grid cells for navigation tasks of a real robot," in *From Animals to Animats 12*, vol. 7426 of *Lecture Notes in Computer Science*, pp. 136–145, Springer, Berlin, Germany, 2012.

[9] https://www.guliverdesign.com/robulab10.

[10] M. J. Milford, J. Wiles, and G. F. Wyeth, "Solving navigational uncertainty using grid cells on robots," *PLoS Computational Biology*, vol. 6, no. 11, Article ID e1000995, 2010.

[11] http://www.mobilerobots.com/ResearchRobots/PioneerP3DX .aspx.

[12] J. I. Cuneo, N. H. Quiroz, V. I. Weisz, and P. F. Argibay, "The computational influence of neurogenesis in the processing of spatial information in the dentate gyrus," *Scientific Reports*, vol. 2, article 735, 2012.

[13] T. Solstad, E. I. Moser, and G. T. Einevoll, "From grid cells to place cells: a mathematical model," *Hippocampus*, vol. 16, no. 12, pp. 1026–1031, 2006.

[14] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[15] https://www.arduino.cc/en/Main/arduinoBoardDue.

[16] H. R. Everett, *Sensors for Mobile Robots: Theory and Application*, A K Peters, Natick, Mass, USA, 1995.

[17] http://www.arducam.com/camera-modules/5mp-ov5642/.

[18] http://arduino-info.wikispaces.com/BlueTooth-HC05-HC06-Modules-How-To.

[19] http://www.st.com/en/motor-drivers/l298.html.

[20] J. Borenstein and L. Feng, "Measurement and correction of systematic odometry errors in mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 6, pp. 869–880, 1996.