# Offset-free MPC of Temperature in Smart Greenhouse VESNA

**Erika Pavlovičová, Jozef Vargan, Peter Bakaráč, Miroslav Fikar, Juraj Oravec**

*Institute of Information Engineering, Automation, and Mathematics, Faculty of Chemical and Food Technology, Slovak University of Technology in Bratislava, Radlinského 9, SK-812 37, Bratislava, e-mail: erika.pavlovicova@stuba.sk.*

**Abstract:** The VESNA smart greenhouse system aims for sustainable, ecological, and organic food production. This study explores an offset-free model predictive controller (MPC) for temperature tracking. The MPC proves effective in maintaining temperature within constraints. Extensive experiments assess different MPC setups, focusing on environmental factors, including energy use and carbon footprint. Additionally, a novel software toolbox simplifies analysis and remote control, enhancing user-friendliness. Together, the designed offset-free reference tracking MPC controllers and the toolbox offer a comprehensive solution for smart greenhouse control.

*Keywords:* Process Control, IoT, Optimization and Model Predictive Control, Smart Greenhouse, Offset-free Control, Object-oriented Programming, Industry 4.0

## 1. INTRODUCTION

Climate change and resource limitations are pushing sustainable production forward. Integrating advanced control technologies, such as smart greenhouses, efficiently addresses these challenges. Applying optimal control methods minimizes the adverse effects on climate (Bersani et al., 2020) and intensifies crop production (Villagran et al., 2020).

Temperature control is crucial in maintaining optimal growing conditions for various crops in greenhouses. Traditional control methods like PID controllers often struggle to achieve precise and efficient temperature performance. Optimization-based control strategies, such as well-known model predictive control (MPC) (Qin and Badgwell, 2003), provide a matured control strategy addressing these limitations and enabling efficient reference tracking temperature control in greenhouse environments. MPC has gained increasing attention as a robust, versatile control method for greenhouse environments. Several studies have investigated the application of MPC in greenhouse temperature control, demonstrating its ability to control temperature effectively while dynamically adapting to changing environmental conditions. An analysis of the control performance of MPC and reinforcement learning for managing a lettuce greenhouse was presented in Morcego et al. (2023). They showed better results using MPC considering energy and economic efficiency. Hu and You (2022) proposed a data-driven robust MPC to control the inner environmental conditions of the greenhouse in the presence of uncertainty.

While the papers above primarily focused on simulated results and theoretical guarantees, the presented paper introduces a comprehensive solution for laboratory greenhouse control, including a user-friendly toolbox for data exchange between the laboratory smart greenhouse VESNA (Versatile Simulator for Near-zero Emissions Agriculture) [1] and the autonomous control unit. Smart greenhouse VESNA and its goals in control and education were introduced in detail in Oravec et al. (2023). This paper investigates the problem of designing and implementing a constrained offset-free MPC for reference tracking of temperature in a smart greenhouse VESNA.

A user-friendly communication toolbox in modern agriculture is indispensable for seamlessly integrating IoT technology with greenhouse devices (Prakash et al., 2023). As part of the paper, a newly developed toolbox `VESNA CODE` [2] is introduced as a user-friendly software for communication with the VESNA greenhouse. The toolbox developed in `MATLAB` programming environment allows the user to easily download and upload data to the greenhouse, allowing the user to stay focused on controller design and fine-tuning of the controller parameters.

## 2. SMART ECO GREENHOUSE VESNA

VESNA, depicted in Figure 1, is a smart ecological greenhouse developed by young researchers and students from the Institute of Informatics, Automation, and Mathematics at FCHPT STU in Bratislava (Oravec et al., 2023). VESNA integrates these two directions into a unique ecosystem, including the greenhouse itself, various sensors, actuators, and a communication interface for data transmission. The sensors collect data on temperature, humidity, light intensity, carbon dioxide concentration, volatile organic compound concentration, and door-opening status. VESNA is also equipped with a heater, an air humidifier, fans, and growth LED strips. The communication

[1] VESNA homepage: `https://vesna.uiam.sk`

[2] `VESNA CODE` homepage: `https://github.com/oravec-juraj/vesna/wiki/How-to-use-VESNA-CODE`

Fig. 1. The smart greenhouse at Slovak University of Technology in Bratislava.

is ensured by a microcontroller, which uses a WiFi module to communicate with `Arduino IoT Cloud`[3] service. A scheme illustrating the closed-loop feedback setup is depicted in Figure 2.

## 3. VESNA CODE

This section introduces the `VESNA CODE` toolbox serving as an advanced human-machine interface (HMI) for building an autonomous control for smart greenhouse VESNA. The `VESNA CODE` interface is developed in the `MATLAB` (MATLAB, 2023) programming environment using a modular and user-friendly object-oriented framework.

### *3.1 VESNA object*

The `VESNA CODE` is built above the object of `vesna`-class, which is created by calling the intuitive `vesna` command:

```
Vesna = vesna
```

An object `Vesna` is created using the constructor and includes a data field with a specific structure. The task of this constructor is to build the default environment of the object, including the initialization of its basic properties:

- `config`,
- `communication`,
- `data`,
- `mail`.

The task of the `config` object property is to store the user-specified configuration data to create a URL request. This request is used to download and upload the data into the remote database. The currently used data storage is the `Arduino IoT Cloud` — an online platform enabling a wide range of data processing (Arduino, 2023). The credentials for accessing this cloud service are stored by the `communication` property. The `data` property stores downloaded data from the `Arduino IoT Cloud`. Settings related to one-way communication between the object and the physical user are contained in the `mail` property. This includes configuration data for sending notifications using some e-mail service.

[3] `Arduino Cloud: https://docs.arduino.cc/arduino-cloud`

### *3.2 Connecting to the cloud*

It is necessary to establish a connection to the server to start the communication of the `vesna` object with the `Arduino IoT Cloud`. The user connects to the remote cloud service by invoking `connect` method

```
Vesna.connect(url, login, password)
```

this notation is based on the so-called *dot*-syntax. Note, the *function*-syntax is also possible, e.g., in the form of: `connect(Vesna, url, login, secret)`.

The `connect` method accepts 3 required input parameters — `url`, `login`, and `password`. The `url` parameter is the public URL address of `Arduino IoT Cloud`, `login` and `password` are private credentials. These data are of type `string` stored in the object property `communication`.

The object `vesna` communicates with the cloud based on API requests, including Bearer authorization containing the Bearer token.

### *3.3 Download data*

Once the connection to the `Arduino IoT Cloud` is established, it is possible to access the data on the cloud. The `download` function is used to download data by calling

```
downloaded_data = Vesna.download(
    variable1, variable2, ...)
```

The arguments `variable1, variable2` are the predefined names of input variables whose data the user wants to access. They are entered in the form of strings, e.g.,

```
downloaded_data = Vesna.download(...
    "temperature", "humidity")
```

From the point of view of the data type, for each variable, the last measured value is downloaded from the cloud by default. Moreover, the corresponding time footprint of the recording is downloaded to prevent handling misleading data.

Reflecting the policy of `Arduino IoT Cloud`, the `download` method allows downloading

- last $n$ data samples,
- arbitrary $m$ data samples, depending on the range of the specified time interval.

### *3.4 Upload data*

Analogous to the data download, the `vesna` object allows uploading user-defined data for actuators (heater, fans, etc.) to the `Arduino IoT Cloud`. The `upload` method is defined for this operation

```
Vesna.upload(variable1,value1,
    variable2,value2,...)
```

where the `upload` function requires entering a pair of data in the form of the variable name (`variable`) and variable value (`value`).

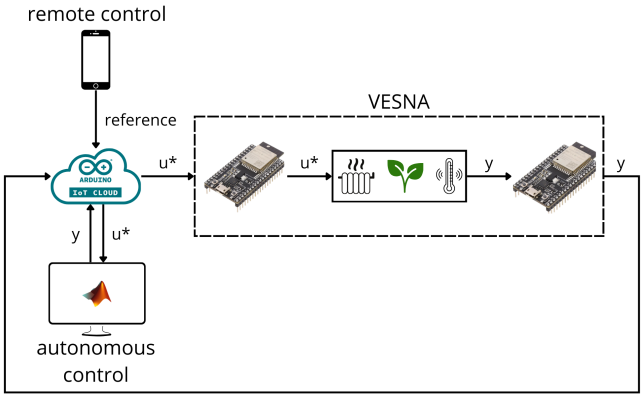


Fig. 2. Scheme of the closed-loop control setup.

## 4. EXPERIMENTAL ANALYSIS

This section presents an extensive analysis of the results of the offset-free MPC control of the inner temperature in the smart greenhouse VESNA. The MPC control was implemented using the early-stage `VESNA CODE` toolbox, providing a user-friendly way of communicating with the device having the potential to supervise advanced control strategies.

### *4.1 Offset-free MPC design*

An experimental analysis aims to control the internal temperature by manipulating the power of the heater, which is located in the lower part of the greenhouse, using MPC. First, to achieve offset-free reference tracking, the discrete-time state-space model for the temperature control in the greenhouse is augmented by an integral action member. Analogous to the well-known LQR with an integral action-based approach, the augmented state-space model can be written in the form

$$\begin{aligned} \widetilde{x}(k+1) &= \widetilde{A}\widetilde{x}(k) + \widetilde{B}u(k), \\ y(k) &= \widetilde{C}\widetilde{x}(k). \end{aligned} \tag{1}$$

where the augmented vector of the system states denoted as $\widetilde{x} \in \mathbb{R}^{n_{\tilde{\mathrm{x}}}}$, is $\widetilde{x}(k) = [x(k), x_{\mathrm{i}}(k)]^\top$, where $x(k) \in \mathbb{R}^{n_{\mathrm{x}}}$ is a vector of state variables and $x_{\mathrm{i}}(k) \in \mathbb{R}^{n_{\mathrm{i}}}$ is an integral action member defined as an integral of a control error. Taking into consideration the augmented vector of the system states $\widetilde{x}$, the matrices of the augmented state-space model are defined as follows

$$\widetilde{A} = \begin{bmatrix} A & 0 \\ -T_s C & I \end{bmatrix} = \begin{bmatrix} 0.7942 & 0 \\ -60 & 1 \end{bmatrix}, \quad \widetilde{B} = \begin{bmatrix} B \\ 0 \end{bmatrix} = \begin{bmatrix} 0.0026 \\ 0 \end{bmatrix}, \tag{2}$$

and $\widetilde{C} = [C\ 0] = [1\ 0]$. The matrices $A$, $B$, and $C$ in (2) are obtained based on the results of an experimental identification. The sampling time $T_s = 60$ s is chosen, considering the practical aspects of the current implementation of the communication with the greenhouse device through the cloud services.

Next, the MPC problem is formulated as the problem of quadratic programming (QP) having the form:

$$\begin{aligned} \min_{u,\widetilde{x}} & \sum_{k=0}^{N-1} \left( \left\| \widetilde{x}(k+1) \right\|_Q^2 + \left\| u(k) \right\|_R^2 \right) \\ \text{s.t. } & \forall k \in \{0, \ldots, N-1\}, \\ & \widetilde{x}(k+1) = \widetilde{A}\widetilde{x}(k) + \widetilde{B}u(k), \\ & u(k) \in \mathbb{U}, \\ & \widetilde{x}(0) = \widetilde{x}_t, \end{aligned} \tag{3}$$

where $N$ stands for the length of a prediction horizon and set $\mathbb{U}$, is a box constraint of control inputs. The matrix $R \in \mathbb{R}^{n_{\mathrm{u}} \times n_{\mathrm{u}}}$ is a penalty matrix of the control inputs $u(k) \in \mathbb{R}^{n_{\mathrm{u}}}$, and $Q \in \mathbb{R}^{n_{\tilde{\mathrm{x}}} \times n_{\tilde{\mathrm{x}}}}$ is a penalty matrix of the system states $\widetilde{x}(k)$. The system states penalty matrix consists of 2 main parts, i.e., the partial matrices $Q_{\mathrm{x}} \in \mathbb{R}^{n_{\mathrm{x}} \times n_{\mathrm{x}}}$ and $Q_{\mathrm{i}} \in \mathbb{R}^{n_{\mathrm{i}} \times n_{\mathrm{i}}}$ penalizing the $x(k)$ and $x_{\mathrm{i}}(k)$), respectively: $Q = \begin{bmatrix} Q_{\mathrm{x}} & 0 \\ 0 & Q_{\mathrm{i}} \end{bmatrix}$.

Within the experiment were investigated several tuning of the penalty matrices, see Table 1. The primary objective in configuring the penalty matrices is to attain optimal control performance, considering appropriate settling speed and minimal energy consumption. The prediction horizon length was constant for all control setups as $N = 20$.

Table 1. Tuning of the penalty matrices.

| Control setup | $Q_{\mathrm{x}}$ | $Q_{\mathrm{i}}$ | $R$ |
|---|---|---|---|
| I | 1 000 | 1 000 | 2.0 |
| II | 1 100 | 1 100 | 1.5 |
| III | 1 000 | 1 000 | 100.0 |
| IV | 1 500 | 1 000 | 1.5 |

To adhere to the physical limitations of the manipulated variable, constraints on the control input $u$ were incorporated in the deviation form $-50 \leq u(k) \leq 50$, corresponding to the operating range of the heater power in the interval $0\,\% - 100\,\%$.

### *4.2 Results and discussion*

Control trajectories obtained by evaluating the MPC [4] for control setups of tuning parameters listed in Table 1 are shown in Figure 3. The control profiles of the temperature in the greenhouse are depicted in the upper figure, while associated control inputs — the power of the heater, are shown below. As can be seen, the constraints on control inputs were active for all Control Setups (Table 1). In contrast to the PID controller, implementing the MPC controller enabled the optimization of the control actions according to these physical limits. On the other hand, implementing the PID controller would require additional saturation of control actions and the necessity to introduce the auxiliary anti-windup control setup. The oscillation observable in the bottom figure related to the profiles of the control actions aims to actively compensate for the impact of the external disturbances on the controlled temperature. As shown in the upper figure, these oscillations are approximately 0.2 °C, hence negligible.

Finally, the achieved control performance was evaluated using several criteria. The settling time refers to when the

[4] Results were obtained using toolbox `YALMIP R20210331` (Löfberg, 2004) and solver `Gurobi 10.0`: `https://www.gurobi.com` in `MATLAB 2020b` programming environment.

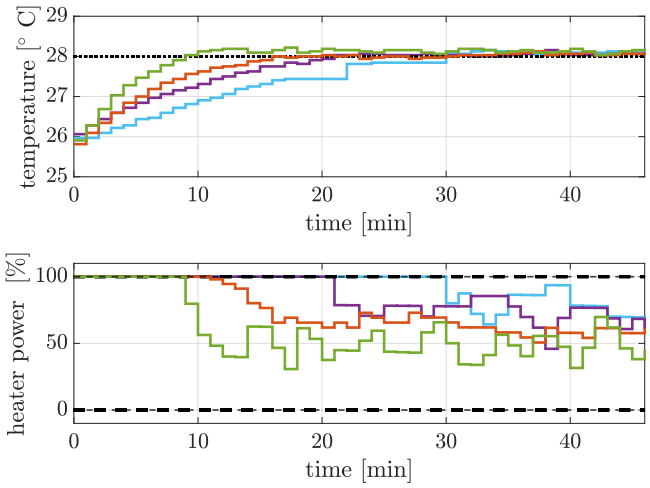


Fig. 3. Control performance ensured by MPC controllers with different tuning of penalty matrices - Control Setup #I (blue solid), Control Setup #II (purple solid), Control Setup #III (orange solid), Control Setup #IV (green solid), reference (black dotted).

controlled variable stabilizes permanently in the neighborhood of the reference variable. The $\delta$-neighborhood of the reference is 0.5 °C, which results from the measurement error of the temperature sensors. Additionally, the overall energy consumption $E$ of the actuator, the heater, was evaluated as $E = T_s \sum_{t=0}^{t_{\text{end}}} Pu(t)$, where $T_s$ is the sampling period in seconds, $u(t)$ is the normalized control input in percent, and $P$ is the power of the heater in Watts. By the term $Pu(t)$, the power in Watts for each control input and in each control step is obtained. Here, the maximum heater's power was 40 W. Analogous to the energy consumption criterion $E$, the corresponding carbon footprint calculated based on the carbon intensity in April 2023 equal to 215 g/kWh [5], when the experimental data were collected.

The MPC controller aimed to reach the reference value fast, without oscillations and steady-state error. As the model of the system was augmented by an integral action, all of the control setups reached the reference value without any steady-state error, except for the negligible impact of the measurement noise (Figure 3). From the point of view of a settling time, the best controller tuning is the Control Setup #IV with increased penalty matrix $Q_{\text{x}}$, shown in the Figure 3 with green solid line as it reached the $\delta$-neighborhood in 6 min. In Figure 3, it can also be seen that the increased penalty matrix $Q_{\text{i}}$ in the Control Setup #II caused more oscillating behavior of the control trajectories, resulting in longer settling time, approximately 12 min. The slowest behavior was related to the Control Setup #I (Figure 3, blue) with the settling time 22 min. However, based on the control profiles of the output variable, it can be assumed that a short-term outage in communication occurred during the experiment.

From the environmental point of view, the highest informative value has the energy consumption of the actuator and the related carbon footprint. The Control Setup #IV provided the minimum energy consumption $E$ equal to 67 kJ as it reached the reference value the fastest, and the control input decreased. Even though the control Control Setup #II provided more conservative control actions, after reaching the reference, the control action dropped but settled around higher values than the previous ones resulting in increased energy consumption 96 kJ. The Control Setup #I with 104 kJ had the highest impact on the environment. However, this may be caused by the already mentioned outage in communication, as the energy consumption and the carbon footprint were computed based on the data obtained from the microcontroller.

[5] Electricity Maps: https://app.electricitymaps.com/map

## ACKNOWLEDGEMENTS

The authors acknowledge Branislav Daráš for his help in realizing the laboratory experiments. The authors gratefully acknowledge the contribution of the Scientific Grant Agency of the Slovak Republic under the grants 1/0297/22, and the Slovak Research and Development Agency under the projects APVV-20-0261, APVV-21-0019. This research is funded by the European Commission under the grant no. 101079342 (Fostering Opportunities Towards Slovak Excellence in Advanced Control for Smart Industries). Erika Pavlovičová and Jozef Vargan thank for a financial contribution from the STU Grant scheme for Support of Young Researchers.

## REFERENCES

Arduino (2023). Cloud Variables. https://docs.arduino.cc/arduino-cloud/cloud-interface/variables. Accessed: 2023-12-06.

Bersani, C., Ouammi, A., Sacile, R., and Zero, E. (2020). Model predictive control of smart greenhouses as the path towards near zero energy consumption. *Energies*, 13(14). doi:10.3390/en13143647.

Hu, G. and You, F. (2022). Model predictive control for greenhouse condition adjustment and crop production prediction. In L. Montastruc and S. Negny (eds.), *32nd European Symposium on Computer Aided Process Engineering*, volume 51 of *Computer Aided Chemical Engineering*, 1051–1056. Elsevier. doi:10.1016/B978-0-323-95879-0.50176-4.

Löfberg, J. (2004). Yalmip : A toolbox for modeling and optimization in MATLAB. In *In Proceedings of the CACSD Conference*. Taipei, Taiwan.

MATLAB (2023). Method Invocation. https://www.mathworks.com/help/matlab/matlab_oop/method-invocation.html. Accessed: 2023-12-05.

Morcego, B., Yin, W., Boersma, S., van Henten, E., Puig, V., and Sun, C. (2023). Reinforcement learning versus model predictive control on greenhouse climate control. *Computers and Electronics in Agriculture*, 215, 108372. doi:10.1016/j.compag.2023.108372.

Oravec, J., Bakaráč, P., Pavlovičová, E., and Fikar, M. (2023). Smart eco greenhouse VESNA. *IFAC-PapersOnLine*, 56(2), 9576–9581. doi:10.1016/j.ifacol.2023.10.260. 22nd IFAC World Congress.

Prakash, C., Singh, L.P., Gupta, A., and Lohan, S.K. (2023). Advancements in smart farming: A comprehensive review of iot, wireless communication, sensors, and hardware for agricultural automation. *Sensors and Actuators A: Physical*, 362, 114605. doi:10.1016/j.sna.2023.114605.

Qin, S.J. and Badgwell, T.A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7), 733–764. doi:10.1016/S0967-0661(02)00186-7.

Villagran, E., Ramirez, R., Rodriguez, A., Pacheco, R.L., and Jaramillo, J. (2020). Simulation of the thermal and aerodynamic behavior of an established screenhouse under warm tropical climate conditions: A numerical approach. *International Journal of Sustainable Development and Planning*, 15, 487–499. doi:10.18280/ijsdp.150409.