

8 Appendix

8.1 Ease of Assembly and Maintenance

The ease of assembly and maintenance is crucial yet difficult to optimize, as it requires mentally simulating the assembly and disassembly process. In early iterations, we explicitly optimize for fewer screw types and ensured unobstructed tool access, which allowed a clear assembly direction for the screwdriver. We also prioritized modular design, allowing individual parts to be replaced independently. These considerations significantly improve maintainability and simplify repairs. As we continue our research on this platform, we have addressed several reliability issues, including motor overheating and link durability. We are committed to releasing ongoing fixes and upgrades to improve the humanoid further.

8.2 Range of Motion

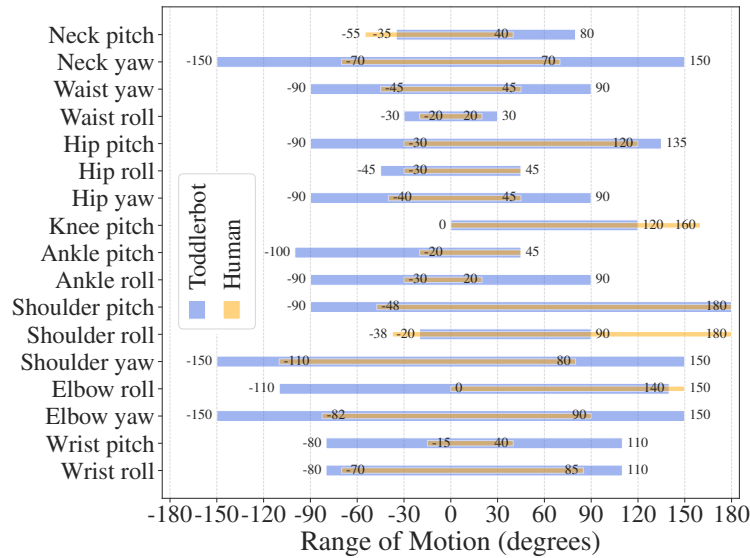


Figure 6: **Range of Motion.** We show that ToddlerBot has near-human or even superhuman mobility in most joints. Negative values represent extensions, adductions, and inversions, while flexions, abductions, and eversion are positive.

We design each joint’s range of motion based on human biomechanics, optimizing geometries to prevent self-collisions and achieve near-human or even superhuman mobility in most joints (Figure 6). The additional extension of the ankle pitch partially compensates for the 40° gap in the knee pitch and the absence of toes. For shoulder roll, although 90° is limited compared to human shoulder abduction, the same hand-up pose can be achieved by actuating the shoulder pitch joint.

8.3 Transmission Mechanisms

Placing motors directly at the joint is often impractical due to the limited space budget. With carefully designed transmission mechanisms, motors can be relocated outside the interference zone, amplify torque output, and offload mechanical stress to the structure. This section highlights key design features, including spur gears, coupled bevel gears, and parallel linkages, as shown in Figure 7.

Each transmission type offers unique benefits. To start with, spur gears provide three advantages:

- **Relocated joint axis:** A 1:1 spur gear set allows repositioning of the joint axis to a more convenient in-plane location. This is widely used in ToddlerBot’s arm.
- **Torque modification:** A ratioed spur gear set adjusts the final torque output, which is particularly useful for the parallel jaw gripper.

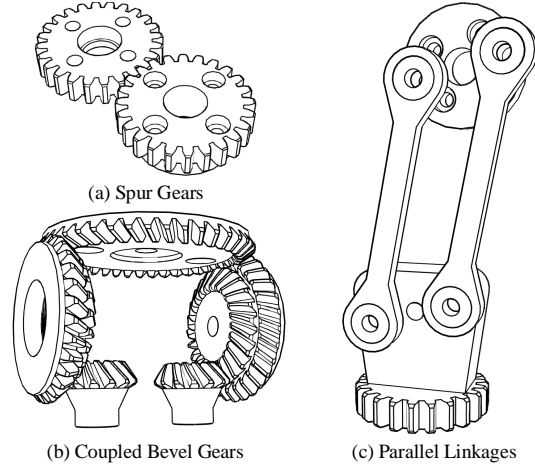


Figure 7: **Transmission Mechanisms.** We show three design primitives in ToddlerBot’s mechanical design: spur gears, coupled bevel gears, and parallel linkages.

456 • **Load distribution:** When a motor’s output shaft has significant free play, as in Dynamixel
 457 XC330, where it is supported only by a Teflon bushing, using it directly as the joint axis is
 458 undesirable. A 1:1 spur gear set enables a reinforced secondary axis with planar bearings and
 459 metal shafts to carry the load, protecting the motor from transverse forces. This approach is used
 460 in the hip yaw joints, where torque demands are low, but load-bearing capacity is critical.

461 With precise tolerance tuning, 3D-printed bevel gears provide a highly interlocking design with
 462 minimal backlash, while still being structurally strong. They also offer three key advantages:

- 463 • **Rotated joint axis:** A coupled bevel gear set enables a parallel waist mechanism, where two
 464 motors in the same orientation drive two perpendicular DoFs.
- 465 • **Combined torque output:** On each axis, both motors contribute to the driving torque, enhancing
 466 power and efficiency. This is critical, as a single Dynamixel XC330 lacks the power to drive the
 467 entire upper body, but two motors combined are sufficient.
- 468 • **Compact actuation:** In the waist, where space is highly constrained, a coupled bevel gear set
 469 allows the compact integration of two DoFs.

470 Lastly, parallel linkages allow the motor to be positioned away from the joint axis, as seen in the knee
 471 and neck pitch. Despite a slightly smaller range of motion (usually $< 160^\circ$), this design efficiently
 472 transfers high torque when paired with ball bearings. They provide three key benefits:

- 473 • **Compact design:** This enables a cleaner neck design by placing the motor inside the head.
- 474 • **Reduced Inertia:** The knee motor is placed higher to reduce rotational inertia.
- 475 • **Structural Efficiency:** In the thigh, the knee motor is bolted to a 3D-printed structure for better
 476 load distribution, increased rigidity, and reduced weight.

477 A potential drawback of these transmission mechanisms is their inaccurate simulation modeling.
 478 However, in MuJoCo [45], we mitigate this by using joint equality constraints for spur gears, fixed
 479 tendons for coupled bevel gears, and weld constraints for parallel linkages. This approach has
 480 empirically shown a small sim2real gap, as demonstrated in Section 5.

481 8.4 Power Factor

482 To quantitatively assess a humanoid robot’s capability, we propose \tilde{p} as the power factor, repre-
 483 senting the total torque (and thus mechanical power) a robot can generate relative to its weight and
 484 height. Intuitively, a higher \tilde{p} means that a humanoid can perform energetic, dynamic motions more

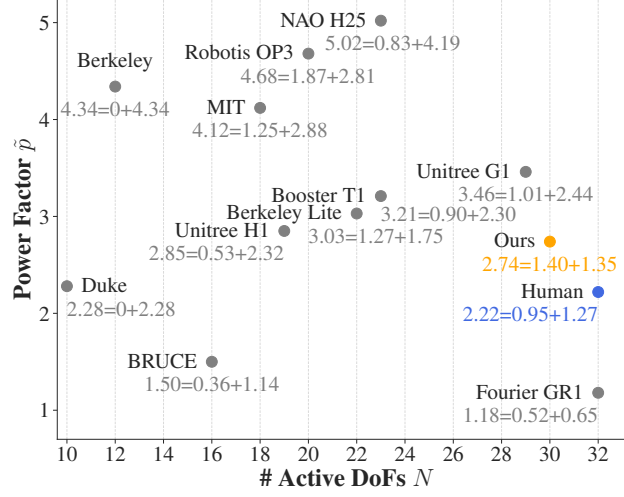


Figure 8: **Humanoid Metrics.** Two key criteria of humanoid capability are the number of active DoFs and power factor \tilde{p} (Equation 4). The total power factor is the sum of the upper and lower body power factor: $\tilde{p}_{total} = \tilde{p}_{upper} + \tilde{p}_{lower}$. ToddlerBot is the closest to human compared with other humanoids, implying potentially comparable loco-manipulation capabilities.

485 easily. We argue that \tilde{p} should at least exceed the human threshold \tilde{p}_{human} to achieve human-like
 486 motion, given the inherent gap between robot and human policies, assuming humans operate as an
 487 oracle policy that is energy efficient. However, raising \tilde{p} far beyond \tilde{p}_{human} can have adverse effects:
 488 unnatural motion, excessive reliance on motor power, fewer DoFs to accommodate larger motors, re-
 489 duced battery life, and increased safety concerns. Thus, pushing \tilde{p} past diminishing returns involves
 490 a practical trade-off. As shown in Figure 8, ToddlerBot has a \tilde{p} score closest to humans.

491 When comparing the performance of humanoids with different scales and weights, directly having
 492 a full-sized 1.8 m humanoid and a 0.5 m scaled humanoid both jump 0.5 m or run at 3 m/s is not
 493 a fair comparison. A more reasonable approach is to normalize performance metrics, for example,
 494 by evaluating a jump at 10% of body height or a running speed of twice the body length per second.
 495 Formally, we say two humanoids have the **same performance** if they execute the same sequence of
 496 joint motions over a time span T , and their total power consumption is the same fraction of their
 497 motors' maximum power:

$$\frac{\int_0^T p(t)dt}{\sum_{i=0}^N |\tau_i^{\max} \dot{q}_i|} \approx \frac{\Delta h \cdot mg}{\sum_{i=0}^N |\tau_i^{\max} \dot{q}_i|} \approx \frac{h \cdot mg}{\sum_{i=0}^N |\tau_i^{\max} \dot{q}_i|}, \quad (3)$$

498 where $p(t)$ is the humanoid's power output at time t . τ_i^{\max} and \dot{q}_i are the maximum torque and joint
 499 velocity of the i -th motor in the humanoid respectively. Moreover, when computing the maximum
 500 power, we take the absolute value to ensure that all motors do positive work. For the numerator,
 501 the integral of the power output over T , which is equivalent to the work done by the robot, can be
 502 approximated by the gravitational energy gained: $\Delta h \cdot mg$. Since Δh is approximately proportional
 503 to the height of the humanoid, we further replace Δh with just the humanoid's height h .

504 Based on Equation 3, we now define power factor to measure the performance of a humanoid:

$$\tilde{p} = \frac{\sum_{i=0}^N |\tau_i^{\max}|}{h \cdot mg}. \quad (4)$$

505 Note that we flip the fraction from Equation 3 to make the power factor value increase as the utilized
 506 torque ratio decreases. We also drop \dot{q} as the same sequence of joint motion would be executed
 507 when using power factor to compare the performance of humanoids.

508 Chi et al. [20] proposes a similar power metric but with an additional normalization by N , the
 509 number of active DoFs. However, consider an extreme case where two humanoids have the same

height and weight: one with a single motor and the other with 100 motors. Normalizing by N would assign them the same power factor, which we believe is inappropriate, as it fails to reflect the true actuation capacity of the system.

8.5 Motor Selection

We choose Dynamixel motors because of their robustness, reliability, and accessibility. Different types of Dynamixel motors were selected for various joints based on space constraints, torque requirements, and cost considerations. In terms of communication speed, Dynamixel motors communicate via a 5V TTL protocol running at 2M baudrate, providing full-state feedback for all 30 motors at 50 Hz using an off-the-shelf communication board. Precision-wise, our motors have a backlash of about 0.25° , on par with most QDD joints used in full-scale humanoids. Also, the Dynamixel encoder resolution is 4096 pulse/rev, and thus the resolution of the motor position reading is 0.09° .

Given ToddlerBot’s size constraint and 30-DoF design, Brushless Direct Drive (BLDC) motors are not a viable option. As BLDC motors shrink, their winding thickness decreases, reducing current capacity and torque constant. Despite their high power density, they still require a high-ratio gearbox, making them less suitable given our limited space budget. We initially explored electric linear actuators, but found them unsuitable due to insufficient power density and low control frequency. After a few iterations, we narrowed our choices to servo motors, ultimately selecting Dynamixel motors for their desirable performance and well-documented support. Given that reproducibility is a hard constraint in our system, we believe a pure Dynamixel design is the most feasible to reproduce, especially for those with limited hardware experience.

For Dynamixel motors, the smallest units start at approximately 50 g. This allows us to estimate the total weight as $3100 = 30 \times 50$ (motors) + 600 (computer, battery, camera) + 1000 (3D-printed structure and metal hardware) g.

To estimate the torque required for each joint to achieve human-like motions, we followed the reference values from [25] and Equation 4 to derive the torque estimation:

$$\tau_{\text{robot}} = \frac{h_{\text{robot}} \cdot m_{\text{robot}}}{h_{\text{human}} \cdot m_{\text{human}}} \cdot \tau_{\text{human}}. \quad (5)$$

With an estimated height of 0.5 m and weight of 3.1 kg, the required lower limb torque for ToddlerBot to perform the most demanding tasks such as running and slope climbing [25], is estimated as follows: $\tau_{\text{robot}}^{\text{knee}} = 2.35$ Nm, $\tau_{\text{robot}}^{\text{ankle pitch}} = 2.66$ Nm, $\tau_{\text{robot}}^{\text{hip pitch}} = 1.77$ Nm.

Table 2: Dynamixel Motor Assignments for ToddlerBot.

Motor Model	Stall Torque ^(a)	Assigned DoFs
XC330-T288	1.0	Neck PY ^(b) , Waist RY, Hip Y, Gripper
XC430-T240BB	1.9	Shoulder P, Ankle R
XM430-W210	3.0	Knee P, Ankle P
2XL430-W250	1.5	Shoulder RY, Elbow RY, Wrist RP
2XC430-W250	1.8	Hip RP

^(a) The Stall Torque data are measured at 12V as reported on the Dynamixel official website [49]. The unit is Nm.

^(b) R, P, and Y denote roll, pitch, and yaw respectively.

As shown in Table 2, XM430 is the only option that provides sufficient torque for the knee and ankle pitch joints. Its metal gears, low backdrive resistance, and high torque output make it ideal for these joints that directly impact walking stability. For the hip, we use 2XC430s for roll and pitch, as they provide sufficient torque while maintaining a compact design, integrating two actuated DoFs in a single housing. This setup allows for a greater range of motion compared to placing two XC430s sequentially. The XC330 series is the smallest and most cost-effective in the lineup, but has higher tracking errors and backlash. Therefore, we use XC330s on joints with lower torque demands or strict weight and space constraints, such as the neck, waist, hip yaw, and parallel jaw

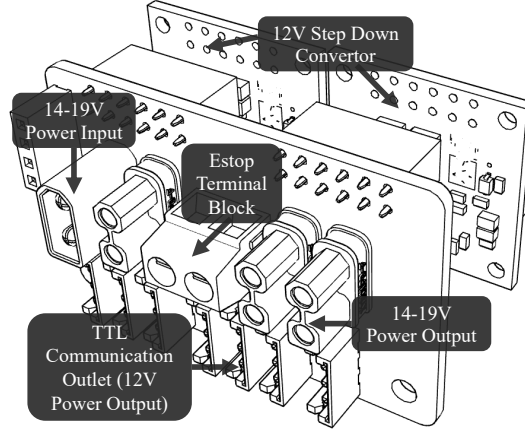


Figure 9: **Power Distribution.** We show the power distribution board design, including four XT30 power plugs, an Estop terminal block, seven JST EH TTL communication outlets, and two 12V step-down convertors.

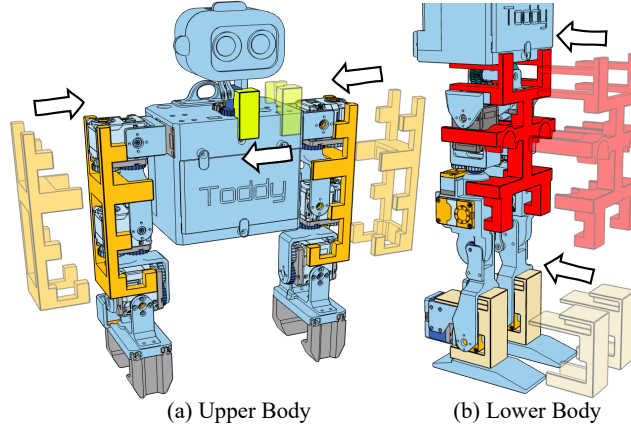


Figure 10: **Zero-point Calibration.** We 3D-print devices for the plug-and-play zero-point calibration procedure: orange for the arm, yellow for the neck, red for the hip, and beige for the ankle. Arrows indicate the insertion direction, and the zero-point is fixed once the devices click into place.

gripper. 2XL430, a lower-cost variant of the 2XC430, is used in the arm to balance performance and cost, ensuring minimal performance loss while maximizing affordability. For the remaining joints, including the shoulder pitch and ankle roll, XC430 is used as a standard choice.

8.6 Power Budget

With the motor selection finalized, we can estimate the power budget. The computer and camera together consume approximately 15 W under typical operating conditions. Actual walking power consumption depends on the energy efficiency of the control policy, but preliminary analysis suggests that the upper body requires minimal power during walking, as it carries no additional load. The lower body operates with alternating support, meaning only one leg is actively working at a time. Assuming a 70% duty cycle, total motor power consumption is estimated at 50 W, requiring a 75 Wh battery for one hour of continuous walking.

For the battery, we offer two options: A 2000 mAh LiPo battery (215 g) available off-the-shelf or a custom-made 4-cell 21700 battery with 5000 mAh, which has higher energy density and weighs 330 g. In practice, the battery lasts 3–5 hours in research settings where the robot walks intermittently. The peak power output from the battery is $14.8 \text{ V} \times 25 \text{ A} = 370 \text{ W}$, sufficient to power all the joints. When debugging without a battery, a 15 V 300 W power supply is a practical alternative—safer and easier to obtain than those required for full-scale humanoids, which often exceed

2 kW and 60 V. In practice, ToddlerBot’s battery can last about 2 hours, longer than other commonly available humanoids - Zeroth [24]: 20 min, OP3 [23]: 10-15 min, BRUCE [21]: 20 min, Berkeley Humanoid Lite [20]: 30 min, NAO H25 [22]: 1 hour, Unitree G1 [15]: 2 hours.

As shown in Figure 9, the battery provides a 14 – 19V input, regulated to 12V via dual step-down converters to power the motors through TTL communication outlets. An E-stop terminal block controls motor power, enabling emergency reboots. The 14 – 19V output powers the Jetson Orin NX, which remains on when the battery is connected to prevent data loss from abrupt shutdown.

8.7 Zero-point Calibration

Figure 10 shows the zero-point calibration process with the 3D-printed devices.

8.8 Motor Test Bed

Each motor family used on ToddlerBot differs in gearbox material, gear ratio, and core, resulting in distinct stiction, backdrive resistance, damping, and inertia. To enable successful sim-to-real transfer, we must minimize the gap between simulated and real actuator behavior.

Specifically, MuJoCo/MJX models a simplified actuator characteristics with predominantly 3 values: frictionloss, damping, and armature [45]. Frictionloss is the minimum torque τ_f required for the actuator to start moving, with a unit of Nm. Damping d refers to the rate at which backdrive resistance increases with speed, measured in Nms/rad. Armature I denotes the effective rotor inertia, accounting for the gearbox, with units of kgm². The model in MuJoCo does not consider stiction peak, therefore, the actuator resistance is simply:

$$\tau_r = \tau_f + d \cdot \dot{q} \quad (6)$$

Therefore, it is straightforward to design a test platform to measure these physical parameters using a series of test sequences. Specifically, damping and friction loss can be measured by backdriving the motor at a constant RPM and recording the resisting torque via a torque sensor. A linear fit to the torque-speed data yields friction loss as the intercept at 0 RPM and damping as the slope. To estimate armature inertia, the actuator is allowed to spin freely before cutting motor power to observe spin-down behavior. Using the previously measured damping values, the resistance power can be numerically integrated to estimate the initial stored energy E . The armature inertia I is then computed using:

$$E = I \cdot \omega^2 \rightarrow I = \frac{E}{\omega^2} \quad (7)$$

We have developed the test profile and analysis code to automatically identify these parameters in MuJoCo. If a more complex actuator model is needed, our modular test bed can be easily extended with additional sensors. The current setup supports active braking (5 Nm), active driving (1 Nm), and accurate torque measurements (precision $3e^{-4}$ Nm), meeting most motor SysID requirements. The test bed will be open-sourced with the release. A photo of the setup is shown in Figure 11.

8.9 Actuation Model

The PD position control equation is computed through:

$$\tau_m = k_p(\hat{q} - q) - (k_d^{\min} + k_d)\dot{q}, \quad (8)$$

where k_p and k_d are the gains, \hat{q} is the joint setpoint, q is the joint position, and \dot{q} is the joint velocity.

During motor testing, we observed significant additional damping when the motor was powered on, even with $k_d = 0$. We modeled this effect as k_d^{\min} , separate from passive damping, as it only occurs when the motor is active and should respect torque limits. Additionally, we found the conversion factor from Dynamixel’s unitless k_p to physical k_p to be approximately 150.

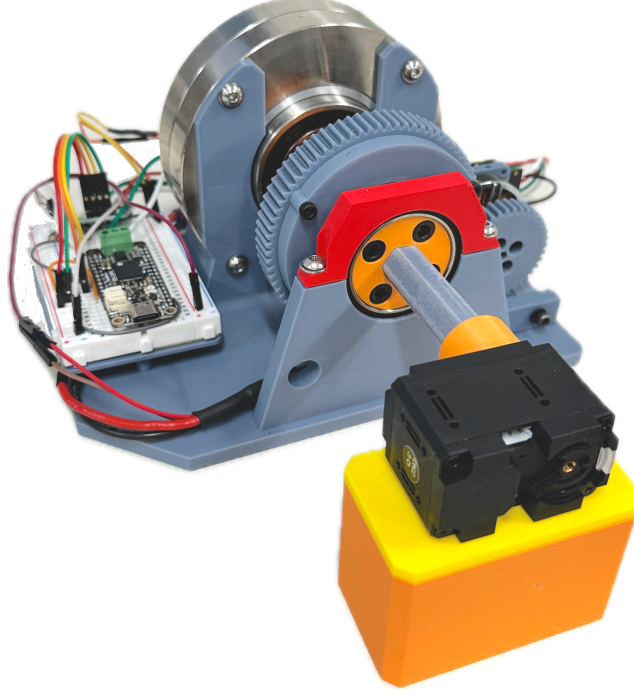


Figure 11: **Motor Test Stand.** We show a photo of the assembled motor test stand. The test motor is mounted via the blue-grey quick-connect shaft, which is directly coupled to a torque sensor. A secondary motor on the side provides active driving torque, while a powder brake at the rear offers up to 5 Nm of controllable resistance. On the left, a controller MCU handles torque sensing, brake actuation, and CAN communication with the driving motor.

Inspired by Grandia et al. [42], we design our actuation model as follows. The motor torque limit τ_{limit} varies with velocity:

$$\tau_{\text{limit}} = \begin{cases} \tau_{\text{max}}, & |\dot{q}| \leq \dot{q}_{\tau_{\text{max}}} \\ \frac{\dot{q}_{\text{max}} - |\dot{q}|}{\dot{q}_{\text{max}} - \dot{q}_{\tau_{\text{max}}}} \cdot \tau_{\text{max}}, & \dot{q}_{\tau_{\text{max}}} < |\dot{q}| \leq \dot{q}_{\text{max}} \\ 0, & |\dot{q}| > \dot{q}_{\text{max}} \end{cases} \quad (9)$$

τ_{limit} comprises a constant torque limit $\tau_{\text{max}} > 0$ for low velocities, and a linear reduction in available torque beyond a specific velocity $\dot{q}_{\tau_{\text{max}}}$. This linear limit reaches zero torque at the velocity \dot{q}_{max} . Note that τ_{limit} is the maximum acceleration torque, but maximum deceleration torque is assumed to be always a constant τ_{brake} . We separate the braking torque limit rather than using the acceleration torque limit as in Grandia et al. [42], since the motor typically provides higher braking torque due to passive resistance and gearbox inefficiencies. The resulting sysID relationship for Dynamixel XC330 is illustrated in Figure 12.

The joint torque is calculated by applying torque limits to τ_m and combining the resistance force τ_r :

$$\tau = \begin{cases} \text{clamp}_{[-\tau_{\text{max}}, \tau_{\text{brake}}]}(\tau_m) - \tau_r, & \dot{q} \geq 0 \\ \text{clamp}_{[-\tau_{\text{brake}}, \tau_{\text{max}}]}(\tau_m) + \tau_r, & \dot{q} < 0 \end{cases} \quad (10)$$

τ_r follows the joint passive force model in MuJoCo [45], which is characterized by three parameters: damping, armature, and friction loss. The sysIDed parameters for various Dynamixel motors are presented in Table 3.

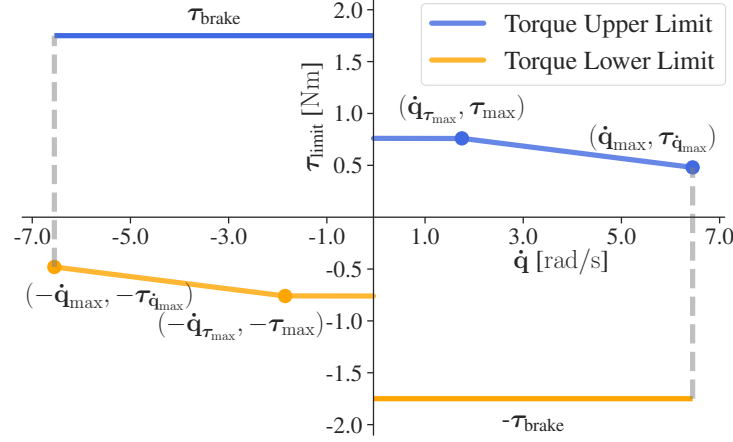


Figure 12: The relationship between torque limit τ_{limit} and joint velocity \dot{q} for Dynamixel XC330.

Table 3: SysIDed Parameters for Dynamixel Motors.

Parameter	2XL430	XC330	XC430	2XC430	XM430-W210
Damping ^(a)	0.0010	0.0036	0.0066	0.0028	0.0056
Armature	0.0083	0.0040	0.0042	0.0044	0.0022
Friction Loss	0.078	0.036	0.024	0.060	0.025
τ_{max}	0.94	0.76	1.32	1.09	1.61
$\dot{q}_{\tau_{\text{max}}}$	2.00	1.80	1.60	2.00	0.10
\dot{q}_{max}	5.97	6.50	7.00	6.78	7.63
$\tau_{\dot{q}_{\text{max}}}$	0.10	0.48	0.21	0.23	0.47
k_d^{min}	0.161	0.384	0.170	0.185	0.203
τ_{brake}	1.40	1.75	3.00	2.20	3.70

^(a) The units of damping and k_d^{min} are Nms/rad, the unit of armature is kgm², the units of frictionloss, τ_{max} and τ_{brake} are Nm, and the units of $\dot{q}_{\tau_{\text{max}}}$ and \dot{q}_{max} are rad/s.

617 Using this actuator model, we jointly optimize all parameters to minimize the gap between simulated
618 and real-world tracking. The model includes 9 parameters: damping, frictionloss, armature, τ_{max} ,
619 $\dot{q}_{\tau_{\text{max}}}$, $\tau_{\dot{q}_{\text{max}}}$, \dot{q}_{max} , k_d^{min} , and τ_{brake} . Due to the complexity of this optimization, constraining
620 parameter ranges is crucial for convergence to a valid local minimum. The final simulation achieves
621 an average tracking error of 1.3° with the optimized parameters (Table 3).

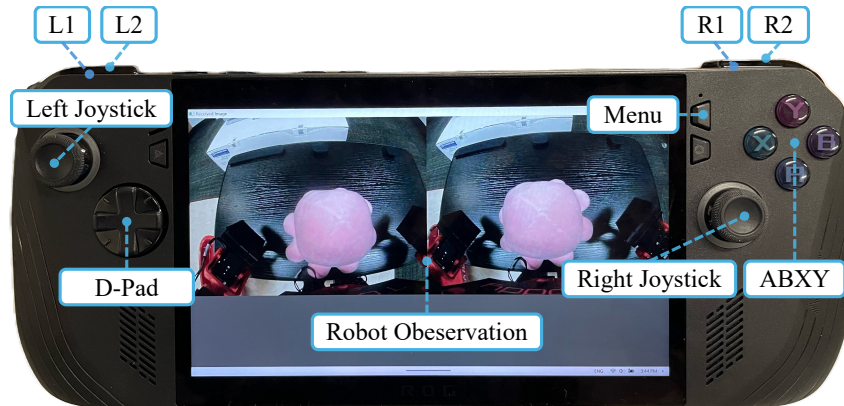


Figure 13: **Remote Controller Layout.** We show the button and axis layout on ROG Ally X.

Table 4: Pupperteering Button and Axis Mapping

Button or Axis ^(a)	Effect
Menu	Toggle teleoperation / Mark episode start or end during data collection
Left Joystick \updownarrow	Walk forward or backward along the x-axis
Left Joystick \leftrightarrow	Walk leftward or rightward along the y-axis
Right Joystick \updownarrow	Stand up or squat down
Right Joystick \leftrightarrow	Turn clockwise or counterclockwise around the z-axis
D-Pad \updownarrow	Lean with the waist roll joint
D-Pad \leftrightarrow	Twist with the waist yaw joint
Y and A	Look up or down with the neck pitch joint
X and B	Look left or right with the neck yaw joint
L1	Hold to run the bimanual DP and release to end.
R1	Hold to run the full-body DP and run to end.
L2	Hold to run the wagon pushing policy and release to end.
R2	Hold to run the cuddling policy and release to end.

^(a) This mapping is compatible with various remote controllers and has been tested on the ROG Ally X and Steam Deck. The remaining buttons can be assigned to additional skills based on user preference.

Table 5: Hyperparameters for PPO Training.

Parameter	Value
Policy hidden layer sizes	(512, 256, 128)
Value hidden layer sizes	(512, 256, 128)
Number of timesteps	3×10^8
Number of environments	1024
Episode length	1000
Unroll length	20
Batch size	256
Number of minibatches	4
Number of updates per batch	4
Discounting factor	0.97
Learning rate	0.0001
Entropy cost	0.0005
Clipping epsilon	0.2

8.10 Pupperteering Mapping

Figure 13 illustrates the remote controller layout to teleoperate ToddlerBot, with button and axis mappings detailed in Table 4. During teleoperation, the human operator sends velocity commands to the walking policy and determines the timing for skill transitions. The same mapping was used for Steam Deck and ROG Ally X.

8.11 Reinforcement Learning Details

The RL implementation leverages MJX [45] and Brax [50]. We train the policy using PPO [46] with hyperparameters listed in Table 5. Inspired by prior work [51, 52], our reward function is shaped by three categories of reward terms as detailed in Table 6. Full implementation details are available

Table 6: Reward Shaping for PPO Training.

Imitation Term	Value
Torso quaternion	1.0
Linear velocity (XY)	5.0
Linear velocity (Z)	1.0
Angular velocity (XY)	2.0
Angular velocity (Z)	5.0
Leg motor position	5.0
Feet contact	1.0
Regularization Term	
Feet air time	500.0
Feet clearance	0.05
Feet distance	1.0
Feet slip	0.05
Align with the ground	1.0
Stand still	1.0
Torso roll	0.5
Torso pitch	0.5
Collision	0.1
Leg action rate	0.05
Leg action acceleration	0.05
Motor torque	0.01
Energy	0.05
Survival Term	
Survival	10.0

in our open-source codebase. During inference, the RL policy runs on the CPU of Jetson Orin NX 16GB, achieving a 50 Hz control loop.

8.12 Diffusion Policy Details

The diffusion policy processes a cropped and downsampled 96×96 RGB image, which is encoded by a ResNet [53] pretrained on ImageNet [54] to extract visual features. Both leader and follower joint angles are downsampled to 10 Hz for training, where the leader joint angles serve as actions and the follower as observations. To prevent motor overload during data collection, the upper body motors use low proportional gains, allowing modulation of the manipulation force. This behavior is embedded in the discrepancy between leader and follower joint angles, which the policy learns.

The model is trained with 100 diffusion steps. During inference, the trained model runs directly on the Jetson Orin NX 16GB with 3 DDPM steps, which are sufficient for satisfactory results. With 300M parameters, the inference latency remains under 0.1 s on the GPU, ensuring smooth execution at 10 Hz without stuttering. Each inference yields a 16-step prediction; the first 3 actions are discarded to compensate for latency [55], and the next 5 actions are executed.

8.13 Velocity Tracking Details

Table 7 shows the quantitative results of the walking velocity tracking experiment.

Table 7: Tracking Errors in Simulation versus the Real World

Tracking Errors	Simulation	Real-World
Position [m]	0.082	0.133 ± 0.018
Linear Velocity [m/s]	0.016	0.032 ± 0.002
Angular Velocity [rad/s]	0.056	0.113 ± 0.010

8.14 Skill Chaining Details

Figure 14 shows the results of the skill chaining experiment.

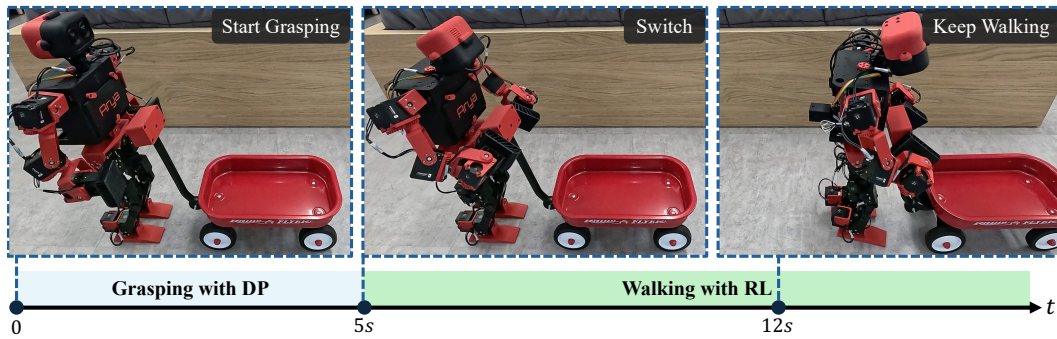


Figure 14: **Skill Chaining.** We demonstrate that ToddlerBot can seamlessly transition from a DP-based grasping skill to an RL-trained walking policy, enabling it to grasp the wagon handle and push it forward while maintaining its grip.

8.15 Reproduction Results

Figure 15 shows some successful reproductions of ToddlerBot by teams around the world.

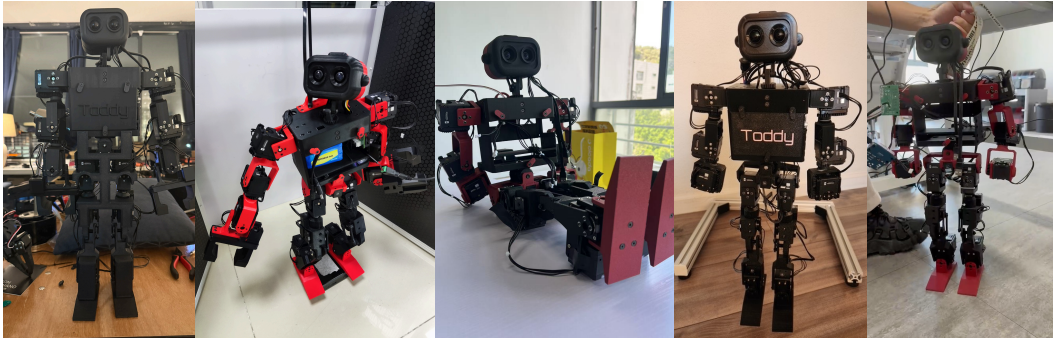


Figure 15: **Successful Reproductions.** We showcase five independent replications of ToddlerBot by teams across the globe. On average, each team completed the replication in about a week.

Figure 16 shows our open-source assembly manual.

